

EXHIBIT A

(12) **United States Patent**
Prahlad et al.

(10) **Patent No.:** **US 7,725,671 B2**
(45) **Date of Patent:** **May 25, 2010**

(54) **SYSTEM AND METHOD FOR PROVIDING
REDUNDANT ACCESS TO METADATA OVER
A NETWORK**

(56) **References Cited**
U.S. PATENT DOCUMENTS

(75) Inventors: **Anand Prahlad**, East Brunswick, NJ
(US); **Jeremy A. Schwartz**, Red Bank,
NJ (US); **David Ngo**, Shrewsbury, NJ
(US); **Brian Brockway**, Shrewsbury, NJ
(US); **Marcus S. Muller**, Tinton Falls,
NJ (US)

4,686,620 A 8/1987 Ng
4,995,035 A 2/1991 Cole et al.
5,005,122 A 4/1991 Griffin et al.
5,093,912 A 3/1992 Dong et al.
5,133,065 A 7/1992 Cheffetz et al.
5,193,154 A 3/1993 Kitajima et al.
5,212,772 A 5/1993 Masters

(73) Assignee: **Comm Vault Systems, Inc.**, Oceanport,
NJ (US)

(Continued)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 711 days.

FOREIGN PATENT DOCUMENTS
EP 0259912 3/1988

(Continued)

(21) Appl. No.: **11/564,194**

OTHER PUBLICATIONS

(22) Filed: **Nov. 28, 2006**

International Search Report and Written Opinion, PCT Application
No. PCT/US2006/61304, Applicant: Commvault Systems, Inc., Date
of Mailing: Mar. 17, 2008, 17 pages.

(65) **Prior Publication Data**

US 2007/0198601 A1 Aug. 23, 2007

(Continued)

Related U.S. Application Data

Primary Examiner—Hiep T Nguyen

(60) Provisional application No. 60/740,686, filed on Nov.
28, 2005, provisional application No. 60/752,203,
filed on Dec. 19, 2005, provisional application No.
60/752,198, filed on Dec. 19, 2005, provisional appli-
cation No. 60/752,196, filed on Dec. 19, 2005, provi-
sional application No. 60/752,202, filed on Dec. 19,
2005, provisional application No. 60/752,201, filed on
Dec. 19, 2005, provisional application No. 60/752,
197, filed on Dec. 19, 2005.

(74) *Attorney, Agent, or Firm*—Perkins Coie LLP

(57) **ABSTRACT**

Systems and methods for data classification to facilitate and
improve data management within an enterprise are described.
The disclosed systems and methods evaluate and define data
management operations based on data characteristics rather
than data location, among other things. Also provided are
methods for generating a data structure of metadata that
describes system data and storage operations. This data struc-
ture may be consulted to determine changes in system data
rather than scanning the data files themselves.

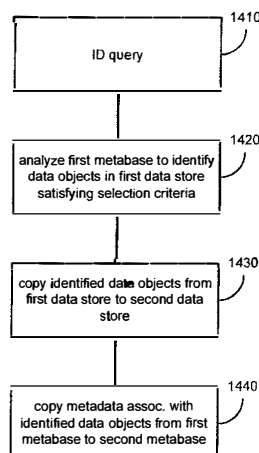
(51) **Int. Cl.**
G06F 12/00 (2006.01)

(52) **U.S. Cl.** **711/162**

(58) **Field of Classification Search** None
See application file for complete search history.

20 Claims, 22 Drawing Sheets

1400



US 7,725,671 B2

Page 2

U.S. PATENT DOCUMENTS

5,226,157	A	7/1993	Nakano et al.	6,418,478	B1	7/2002	Ignatius et al.
5,239,647	A	8/1993	Anglin et al.	6,421,683	B1	7/2002	Lamburt
5,241,668	A	8/1993	Eastridge et al.	6,421,711	B1	7/2002	Blumenau et al.
5,241,670	A	8/1993	Eastridge et al.	6,421,779	B1	7/2002	Kuroda et al.
5,276,860	A	1/1994	Fortier et al.	6,430,575	B1	8/2002	Dourish et al.
5,276,867	A	1/1994	Kenley et al.	6,438,586	B1	8/2002	Hass et al.
5,287,500	A	2/1994	Stoppani, Jr.	6,487,561	B1	11/2002	●fek et al.
5,321,816	A	6/1994	Rogan et al.	6,487,644	B1	11/2002	Huebsch et al.
5,333,315	A	7/1994	Saether et al.	6,519,679	B2	2/2003	Devireddy et al.
5,347,653	A	9/1994	Flynn et al.	6,538,669	B1	3/2003	Lagueux, Jr. et al.
5,410,700	A	4/1995	Fecteau et al.	6,542,909	B1	4/2003	Tamer et al.
5,448,724	A	9/1995	Hayashi et al.	6,542,972	B2	4/2003	Ignatius et al.
5,491,810	A	2/1996	Allen	6,564,228	B1	5/2003	●'Connor
5,495,607	A	2/1996	Pisello et al.	6,581,143	B2	6/2003	Gagne et al.
5,504,873	A	4/1996	Martin et al.	6,625,623	B1	9/2003	Midgley et al.
5,519,865	A	5/1996	Kondo et al.	6,658,436	B2	12/2003	●shinsky et al.
5,544,345	A	8/1996	Carpenter et al.	6,658,526	B2	12/2003	Nguyen et al.
5,544,347	A	8/1996	Yanai et al.	6,732,124	B1	5/2004	Koseki et al.
5,559,957	A	9/1996	Balk	6,772,164	B2	8/2004	Reinhardt
5,619,644	A	4/1997	Crockett et al.	6,775,790	B2	8/2004	Reuter et al.
5,638,509	A	6/1997	Dunphy et al.	6,836,779	B2	12/2004	Poulin
5,673,381	A	9/1997	Huai et al.	6,847,984	B1	1/2005	Midgley et al.
5,699,361	A	12/1997	Ding et al.	6,886,020	B1	4/2005	Zahavi et al.
5,729,743	A	3/1998	Squibb	6,947,935	B1	9/2005	Horvitz et al.
5,737,747	A	4/1998	Vishlitzky et al.	6,983,322	B1	1/2006	Tripp et al.
5,751,997	A	5/1998	Kullick et al.	6,996,616	B1	2/2006	Leighton et al.
5,758,359	A	5/1998	Saxon	7,035,880	B1	4/2006	Crescenti et al.
5,761,677	A	6/1998	Senator et al.	7,047,236	B2	5/2006	Conroy et al.
5,764,972	A	6/1998	Crouse et al.	7,085,787	B2	8/2006	Beier et al.
5,778,395	A	7/1998	Whiting et al.	7,103,740	B1	9/2006	Colgrove et al.
5,812,398	A	9/1998	Nielsen	7,130,860	B2	10/2006	Pachet et al.
5,813,009	A	9/1998	Johnson et al.	7,130,970	B2	10/2006	Devassy et al.
5,813,017	A	9/1998	Morris	7,167,895	B1	1/2007	Connelly
5,829,046	A	10/1998	Tzelnic et al.	7,181,444	B2	2/2007	Porter et al.
5,832,510	A	11/1998	Ito et al.	7,197,502	B2	3/2007	Feinsmith
5,875,478	A	2/1999	Blumenau	7,240,100	B1	7/2007	Wein et al.
5,887,134	A	3/1999	Ebrahim	7,246,207	B2	7/2007	Kottomtharayil et al.
5,892,917	A	4/1999	Myerson	7,246,211	B1	7/2007	Belousov et al.
5,901,327	A	5/1999	●fek	7,330,997	B1	2/2008	●dom
5,907,621	A	5/1999	Bachman et al.	7,346,623	B2	3/2008	Prahlad et al.
5,924,102	A	7/1999	Perks	7,346,676	B1	3/2008	Swildens et al.
5,950,205	A	9/1999	Aviani, Jr.	7,356,660	B2	4/2008	Matsunami et al.
5,953,721	A	9/1999	Doi et al.	7,386,663	B2	6/2008	Cousins
5,974,563	A	10/1999	Beeler, Jr.	7,440,966	B2	10/2008	Adkins et al.
6,021,415	A	2/2000	Cannon et al.	7,454,569	B2	11/2008	Kavuri et al.
6,026,414	A	2/2000	Anglin	7,533,103	B2	5/2009	Brendle et al.
6,052,735	A	4/2000	Ulrich et al.	7,583,861	B2	9/2009	Hanna et al.
6,061,692	A	5/2000	Thomas et al.	7,590,997	B2	9/2009	Diaz Perez
6,076,148	A	6/2000	Kedem et al.	2002/0069324	A1	6/2002	Gerasimov et al.
6,094,416	A	7/2000	Ying	2002/0087550	A1	7/2002	Carlyle et al.
6,131,095	A	10/2000	Low et al.	2003/0018607	A1	1/2003	Lennon et al.
6,131,190	A	10/2000	Sidwell	2003/0115219	A1	6/2003	Chadwick
6,148,412	A	11/2000	Cannon et al.	2003/0130993	A1	7/2003	Mendelevitch et al.
6,154,787	A	11/2000	Urevig et al.	2003/0182583	A1	9/2003	Turco
6,154,852	A	11/2000	Amundson et al.	2004/0010493	A1	1/2004	Kojima et al.
6,161,111	A	12/2000	Mutalik et al.	2004/0015514	A1	1/2004	Melton et al.
6,167,402	A	12/2000	Yeager	2004/0254919	A1	12/2004	Giuseppini
6,212,512	B1	4/2001	Barney et al.	2004/0260678	A1	12/2004	Verbowski et al.
6,260,069	B1	7/2001	Anglin	2005/0050075	A1	3/2005	●kamoto et al.
6,269,431	B1	7/2001	Dunham	2005/0114406	A1	5/2005	Borthakur et al.
6,275,953	B1	8/2001	Vahalia et al.	2005/0154695	A1	7/2005	Gonzalez et al.
6,301,592	B1	10/2001	Aoyama et al.	2005/0188248	A1	8/2005	●'Brien et al.
6,324,581	B1	11/2001	Xu et al.	2005/0193128	A1	9/2005	Dawson et al.
6,328,766	B1	12/2001	Long	2005/0216453	A1	9/2005	Sasaki et al.
6,330,570	B1	12/2001	Crighton et al.	2005/0228794	A1	10/2005	Navas et al.
6,330,642	B1	12/2001	Carteau	2005/0262097	A1	11/2005	Sim-Tang et al.
6,343,324	B1	1/2002	Hubis et al.	2006/0004820	A1	1/2006	Claudatos et al.
RE37,601	E	3/2002	Eastridge et al.	2006/0010227	A1	1/2006	Atluri
6,356,801	B1	3/2002	Goodman et al.	2006/0031225	A1	2/2006	Palmeri et al.
6,374,336	B1	4/2002	Peters et al.	2006/0031263	A1	2/2006	Arrouye et al.
6,389,432	B1	5/2002	Pothapragada et al.	2006/0101285	A1	5/2006	Chen et al.
				2006/0106814	A1	5/2006	Blumenau et al.
				2006/0253495	A1	11/2006	Png

US 7,725,671 B2

Page 3

2006/0259468 A1 11/2006 Brooks et al.
 2006/0294094 A1 12/2006 King et al.
 2007/0033191 A1 2/2007 Hornkvist et al.
 2007/0112809 A1 5/2007 Arrouye et al.
 2008/0021921 A1 1/2008 Horn
 2008/0091655 A1 4/2008 Gokhale et al.

FOREIGN PATENT DOCUMENTS

EP 0405926 1/1991
 EP 0467546 1/1992
 EP 0774715 5/1997
 EP 0809184 11/1997
 EP 0899662 3/1999
 EP 0981090 2/2000
 EP 1174795 1/2002
 WO WO-95/13580 5/1995
 WO WO-99/12098 3/1999
 WO WO-99/14692 3/1999
 WO WO-03060774 7/2003
 WO WO-2005/055093 6/2005
 WO WO-2007/062254 5/2007
 WO WO-2007/062254 5/2007
 WO WO-2007/062429 5/2007
 WO WO-2008/049023 4/2008

OTHER PUBLICATIONS

EMC Corporation, "Today's Choices for Business Continuity," 2004, 12 pages.
 Microsoft Developer Network, "Win32_File_Attribute_Data," online library article, [accessed on Nov. 10, 2005], 3 pages.
 Microsoft Developer Network, "GetFileAttributesEx," online library article, [accessed on Nov. 10, 2005], 2 pages.
 Microsoft Developer Network, "GetFileAttributes," online library article, [accessed on Nov. 10, 2005], 3 pages.
 Jeffrey Richter and Luis Felipe Cabrera, "A File System for the 21st Century: Previewing the Windows NT 5.0 File System," and attached text figures, Microsoft Systems Journal, Nov. 1998, 24 pages.
 Jeffrey Cooperstein and Jeffrey Richter, "Keeping an Eye on Your NTFS Drives: the Windows 2000 Change Journal Explained," Microsoft Systems Journal, Sep. 1999, 17 pages.
 Jeffrey Cooperstein and Jeffrey Richter, "Keeping an Eye on Your NTFS Drives, Part II: Building a Change Journal Application," Microsoft Systems Journal, Oct. 1999, 14 pages.
 Brad Neill, "New Tools to Classify Data," Storage Magazine, Aug. 2005, 4 pages.
 Karl Langdon and John Merryman, "Data Classification: Getting Started," Storage Magazine, Jul. 2005, 3 pages.
 Non-Final Office Action for U.S. Appl. No. 11/564,153, Mail Date Nov. 14, 2008, 22 pages.
 Final Office Action for U.S. Appl. No. 11/931,034, Mail Date Dec. 29, 2008, 18 pages.
 Supplementary European Search Report for European Application EP06846386, Dated Dec. 30, 2008, European Patent Office, 6 pages.
 European Patent Office Examination Report for EP application 06846386.8, Mail Date Apr. 29, 2009, 6 pages.
 Non-Final Office Action for U.S. Appl. No. 11/564,215, Mail Date May 8, 2009, 39 pages.
 Non-Final Office Action for U.S. Appl. No. 11/564,136, Mail Date May 15, 2009, 25 pages.
 U.S. Appl. No. 11/931,034, filed Oct. 31, 2007, Kottomtharayil et al.

U.S. Appl. No. 12/058,487, filed Mar. 28, 2008, Prahlad.
 U.S. Appl. No. 12/058,575, filed Mar. 28, 2008, Prahlad et al.
 U.S. Appl. No. 12/058,589, filed Mar. 28, 2008, Prahlad et al.
 Arneson, David A., "Development of Omniser," Control Data Corporation, Tenth IEEE Symposium on Mass Storage Systems, May 1990, 'Crisis in Mass Storage' Digest of Papers, pp. 88-93, Monterey, CA.
 PCT International Search Report and Written Opinion for International Application No. PCT/US07/81681, Mail Date Oct. 20, 2008, 11 pages.
 U.S. Appl. No. 11/563,940, Prahlad et al.
 U.S. Appl. No. 11/564,119, Prahlad et al.
 U.S. Appl. No. 11/564,130, Prahlad et al.
 U.S. Appl. No. 11/564,136, Prahlad et al.
 U.S. Appl. No. 11/564,153, Prahlad et al.
 U.S. Appl. No. 11/564,163, Prahlad et al.
 U.S. Appl. No. 11/564,170, Prahlad et al.
 U.S. Appl. No. 11/564,174, Prahlad et al.
 U.S. Appl. No. 11/564,177, Prahlad et al.
 U.S. Appl. No. 11/564,180, Prahlad et al.
 U.S. Appl. No. 11/564,197, Prahlad et al.
 U.S. Appl. No. 11/564,215, Prahlad et al.
 U.S. Appl. No. 11/564,220, Prahlad et al.
 U.S. Appl. No. 11/564,221, Prahlad et al.
 U.S. Appl. No. 11/564,233, Prahlad et al.
 U.S. Appl. No. 11/605,931, Prahlad et al.
 U.S. Appl. No. 11/605,932, Prahlad et al.
 U.S. Appl. No. 11/605,944, Prahlad et al.
 Armstead et al., "Implementation of a Campus-wide Distributed Mass Storage Service: The Dream vs. Reality," IEEE, 1995, pp. 190-199.
 Arneson, "Mass Storage Archiving in Network Environments," Digest of Papers, Ninth IEEE Symposium on Mass Storage Systems, Oct. 31, 1988-Nov. 3, 1988, pp. 45-50, Monterey, CA.
 Cabrera et al., "ADSM: A Multi-Platform, Scalable, Backup and Archive Mass Storage System," Digest of Papers, Comcon '95, Proceedings of the 40th IEEE Computer Society International Conference, Mar. 5, 1995-Mar. 9, 1995, pp. 420-427, San Francisco, CA.
 Eitel, "Backup and Storage Management in Distributed Heterogeneous Environments," IEEE, 1994, pp. 124-126.
 Jander, M., "Launching Storage-Area Net," Data Communications, US, McGraw Hill, NY, vol. 27, No. 4 (Mar. 21, 1998), pp. 64-72.
 Jason Gait, "The Optical File Cabinet: A Random-Access File System For Write-Once Optical Disks," IEEE Computer, vol. 21, No. 6, pp. 11-22 (1988) (see in particular figure 5 in p. 15 and recitation in claim 5).
 Rosenblum et al., "The Design and Implementation of a Log-Structured File System," Operating Systems Review SIGOPS, vol. 25, No. 5, New York, US, pp. 1-15 (May 1991).
 Partial International Search Results, mailed May 25, 2007, International Application No. PCT/US2006/045556, 2 pages.
 "Text Figures", retrieved from <http://www.microsoft.com/msj/1198.ntfs/ntfstextfigs.htm> on Nov. 10, 2005, 7 pages.
 U.S. Appl. No. 12/548,953, filed Aug. 27, 2009, Ahn et al.
 International Search Report and Written Opinion for International Application No. PCT/US07/81681, Mail Date Nov. 13, 2009, 8 pages.
 U.S. Appl. No. 12/511,653, filed Jul. 29, 2009, Prahlad et al.
 Communication with extended European Search Report for Application No. PCT/US2006/061304, dated Dec. 30, 2008.

100

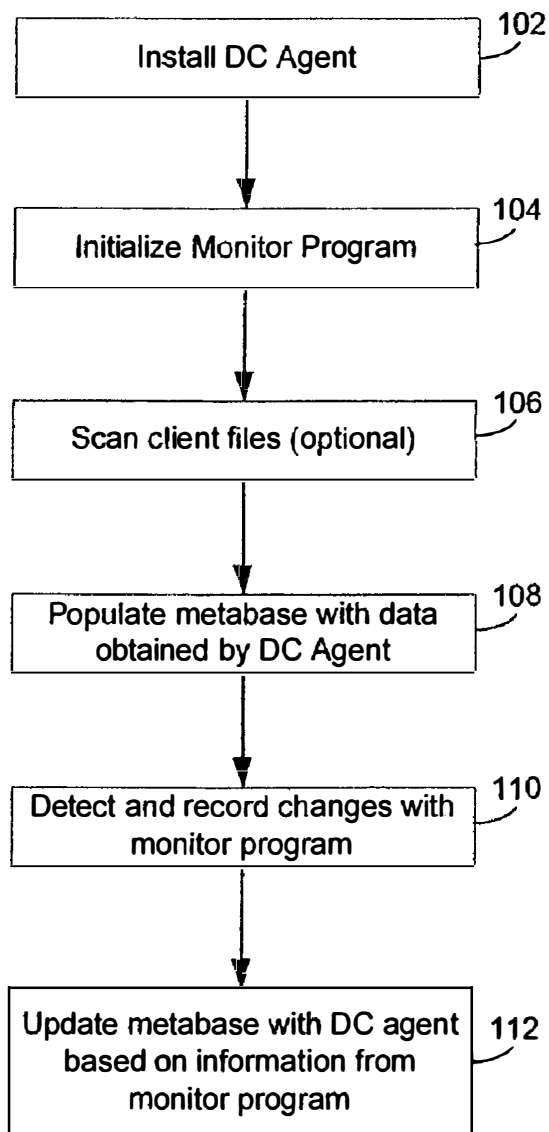


FIG. 1

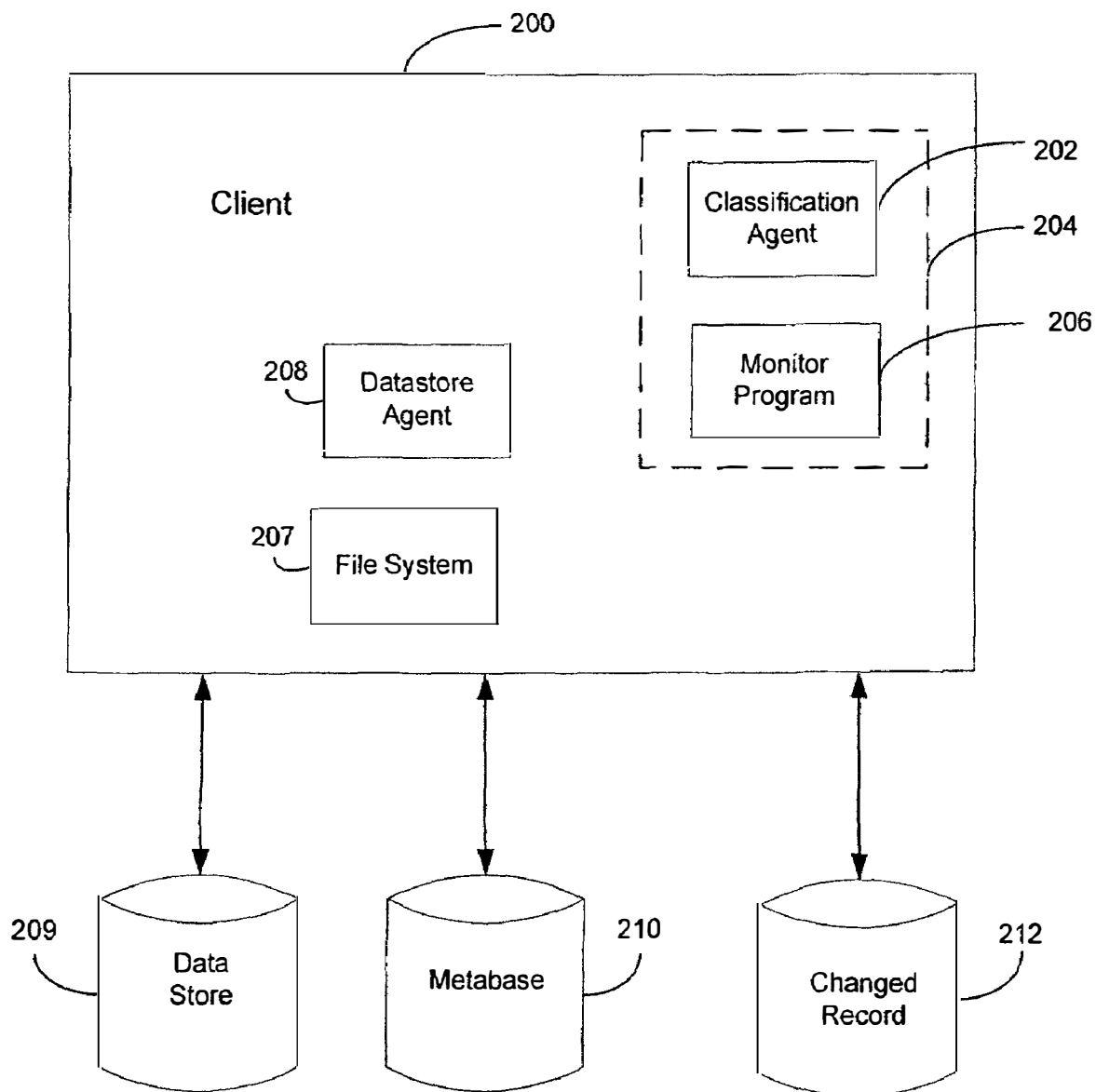


FIG. 2

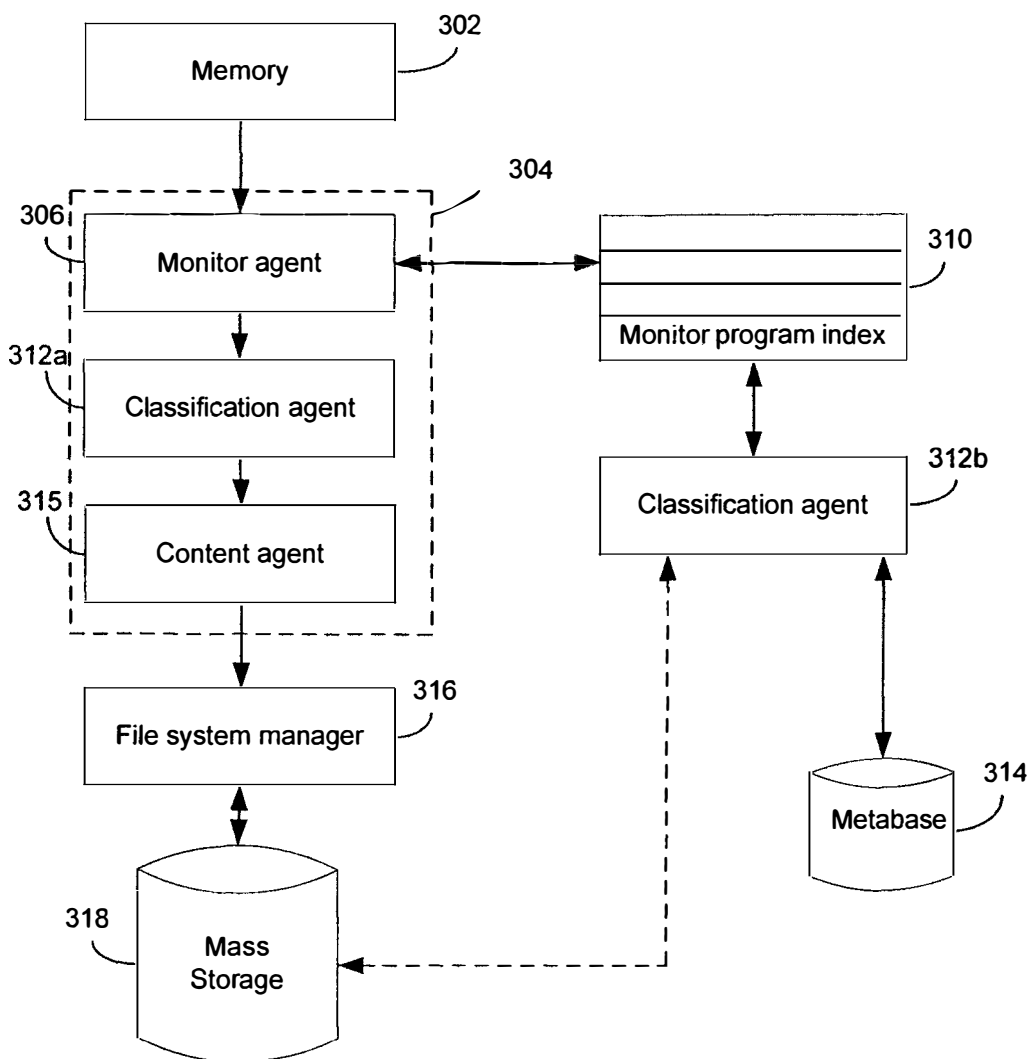
300

FIG. 3a

350

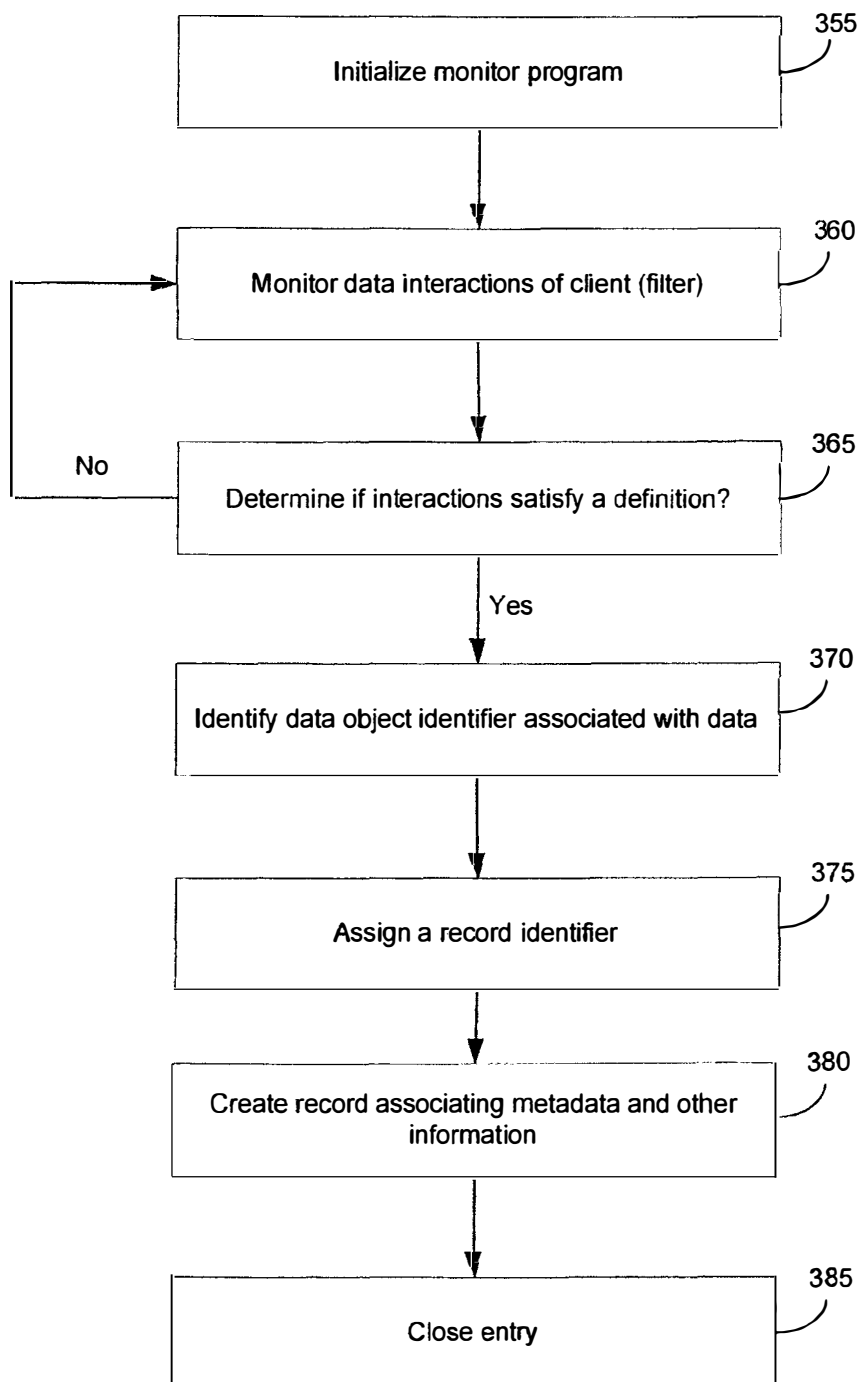


FIG. 3b

400

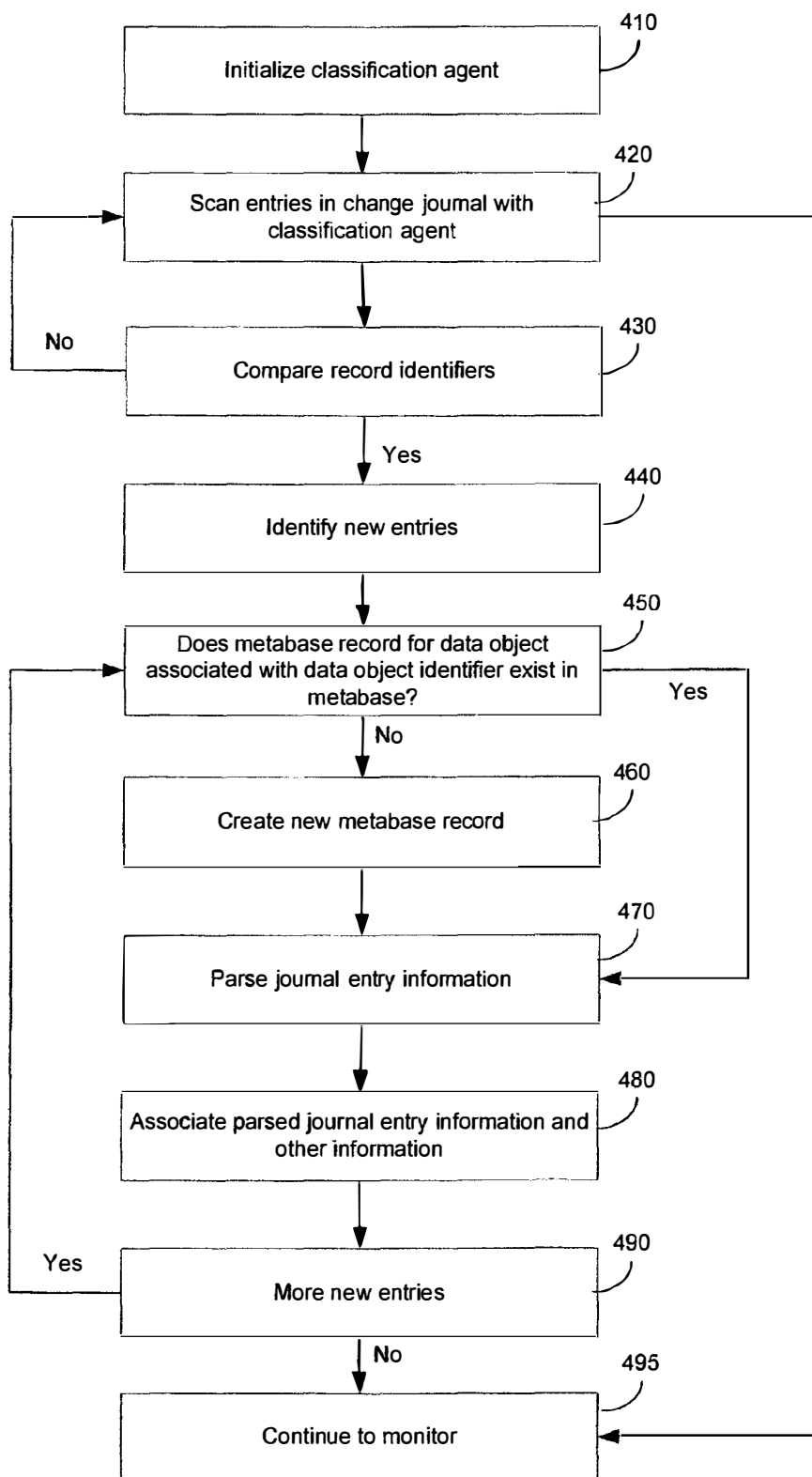


FIG. 4

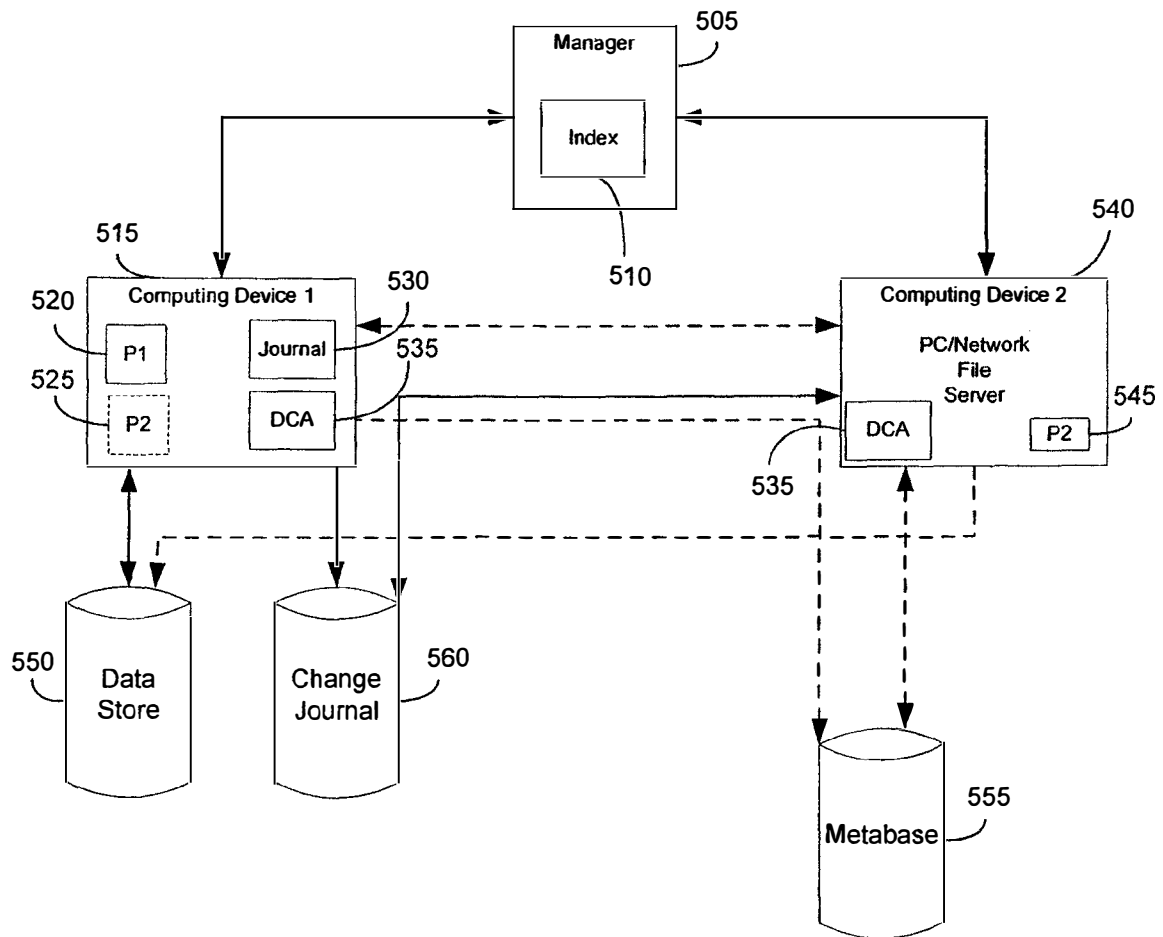
500

FIG. 5

600

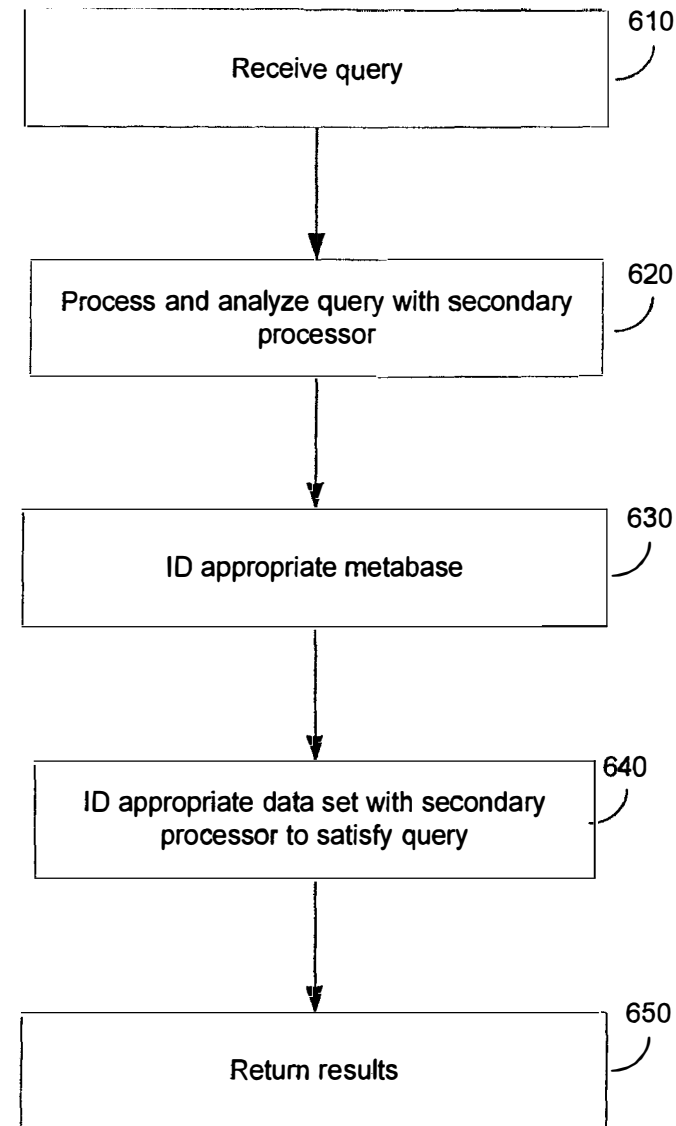


FIG. 6

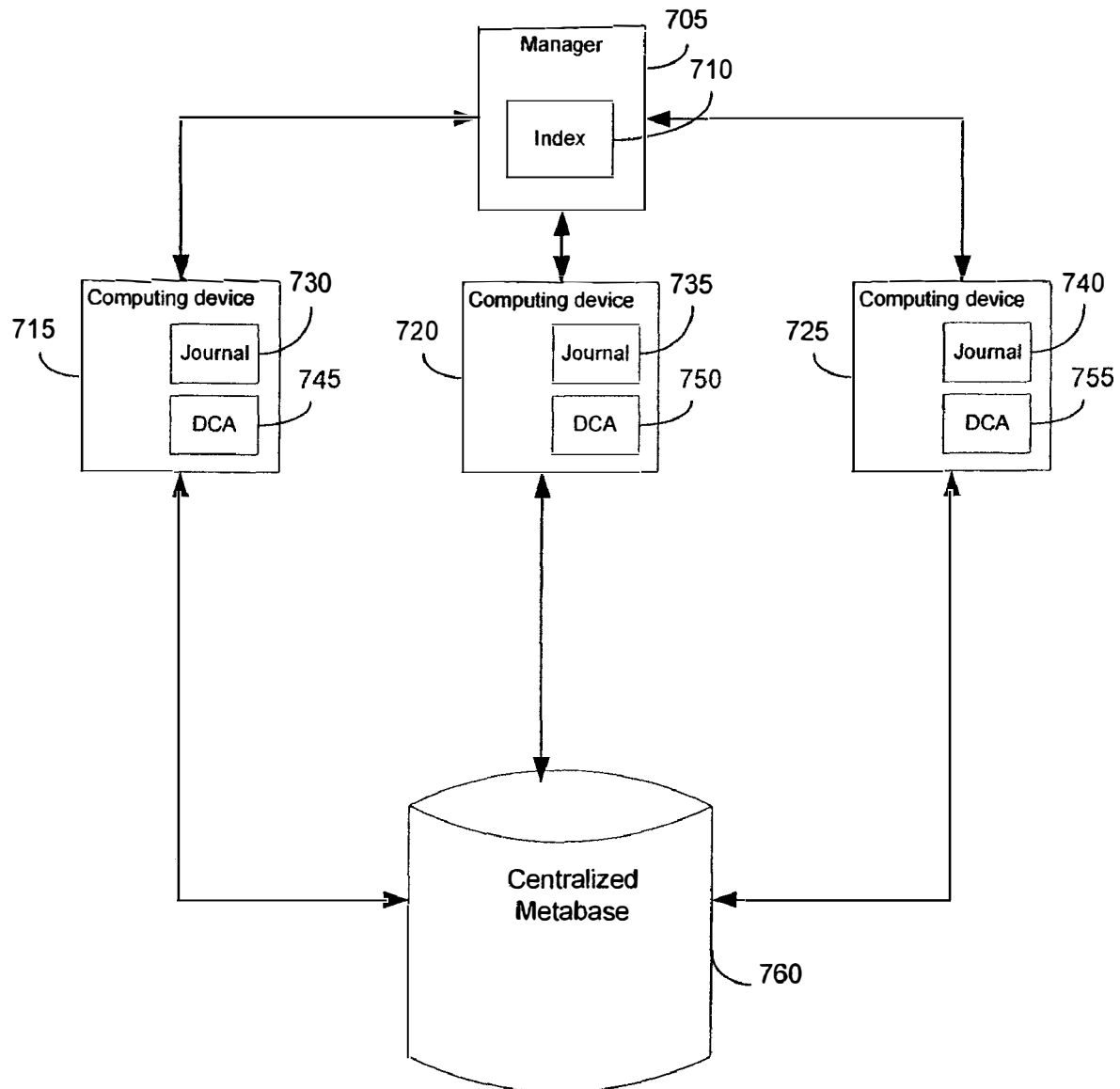
700

FIG. 7

800

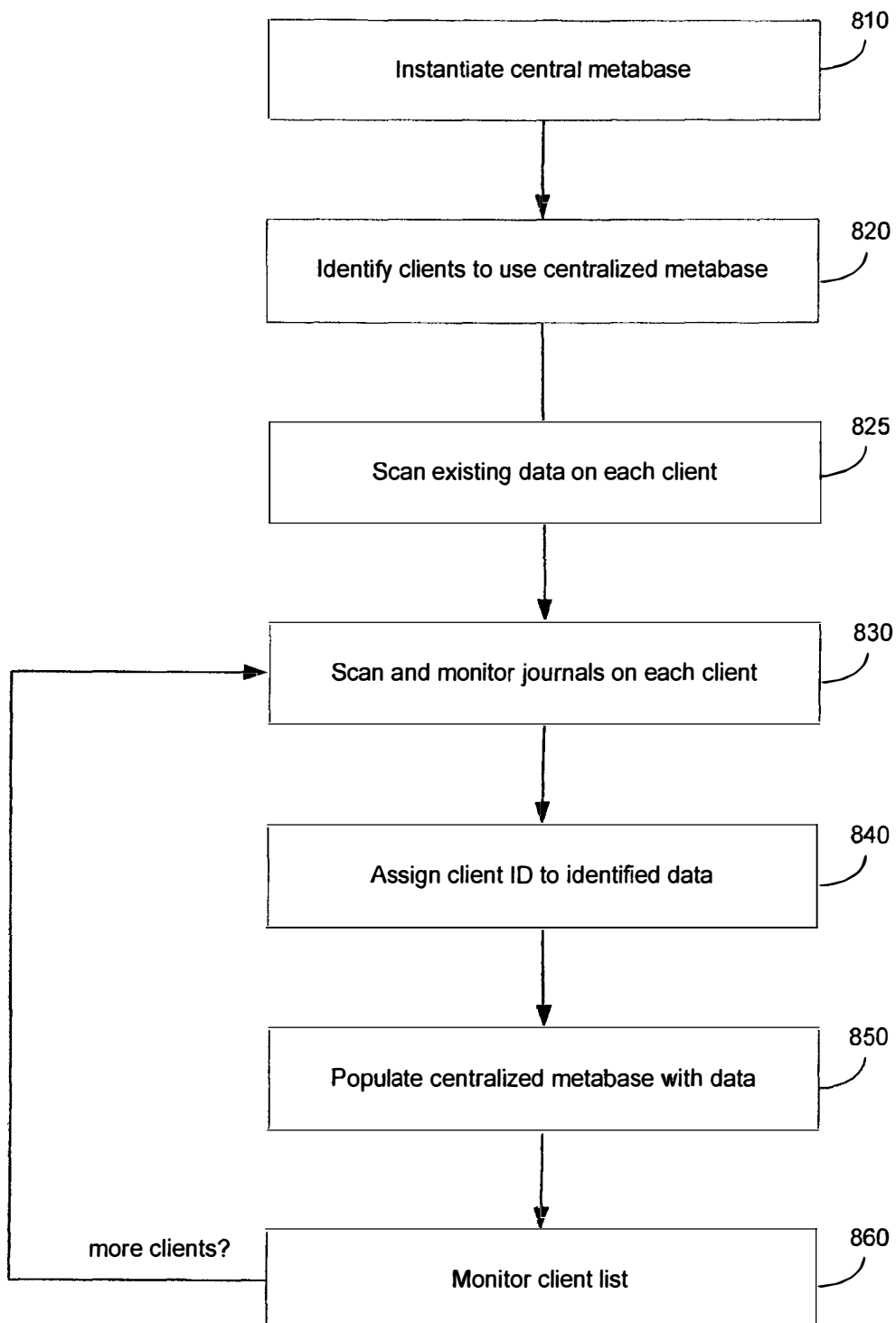


FIG. 8

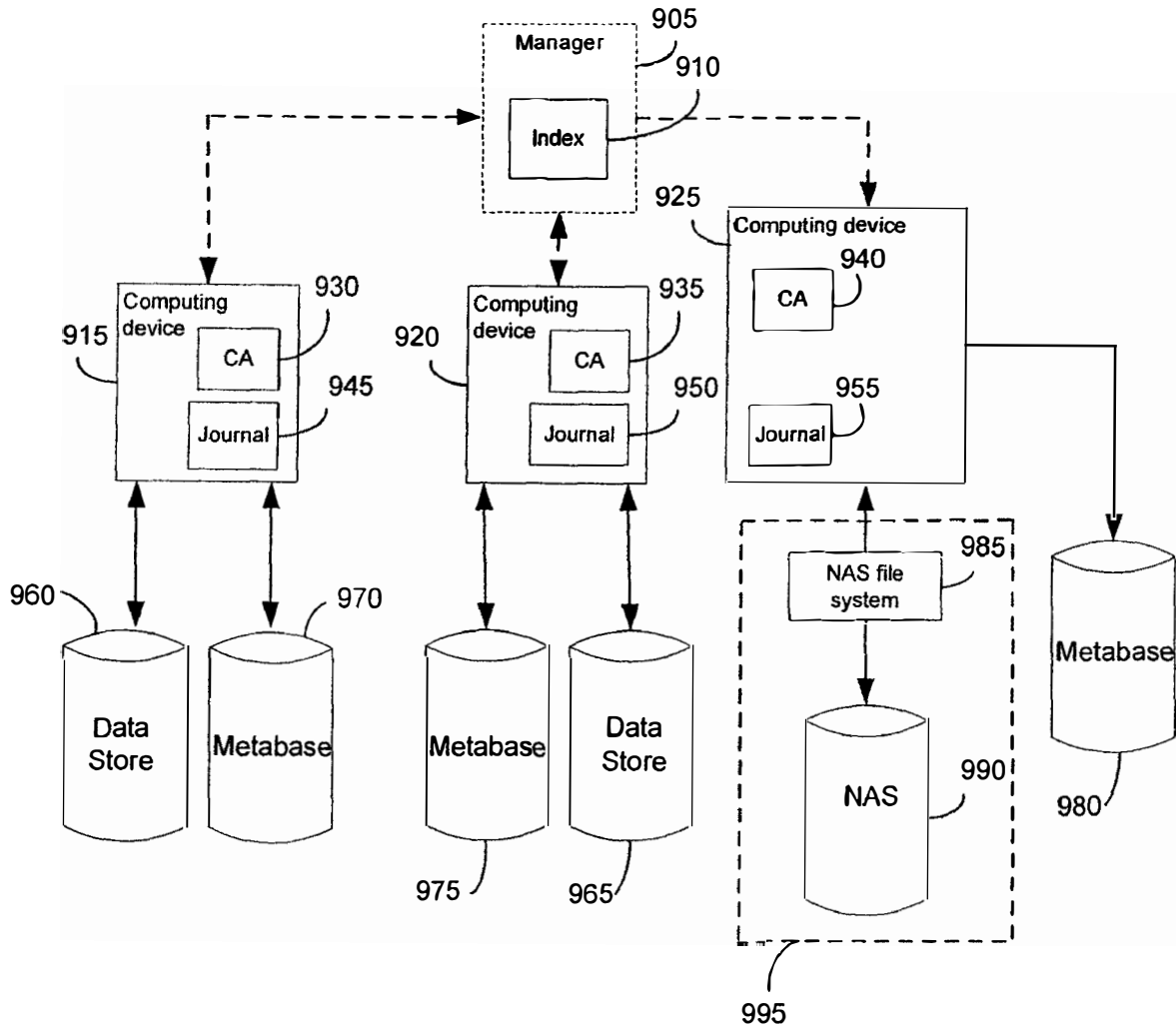
900

FIG. 9

1000

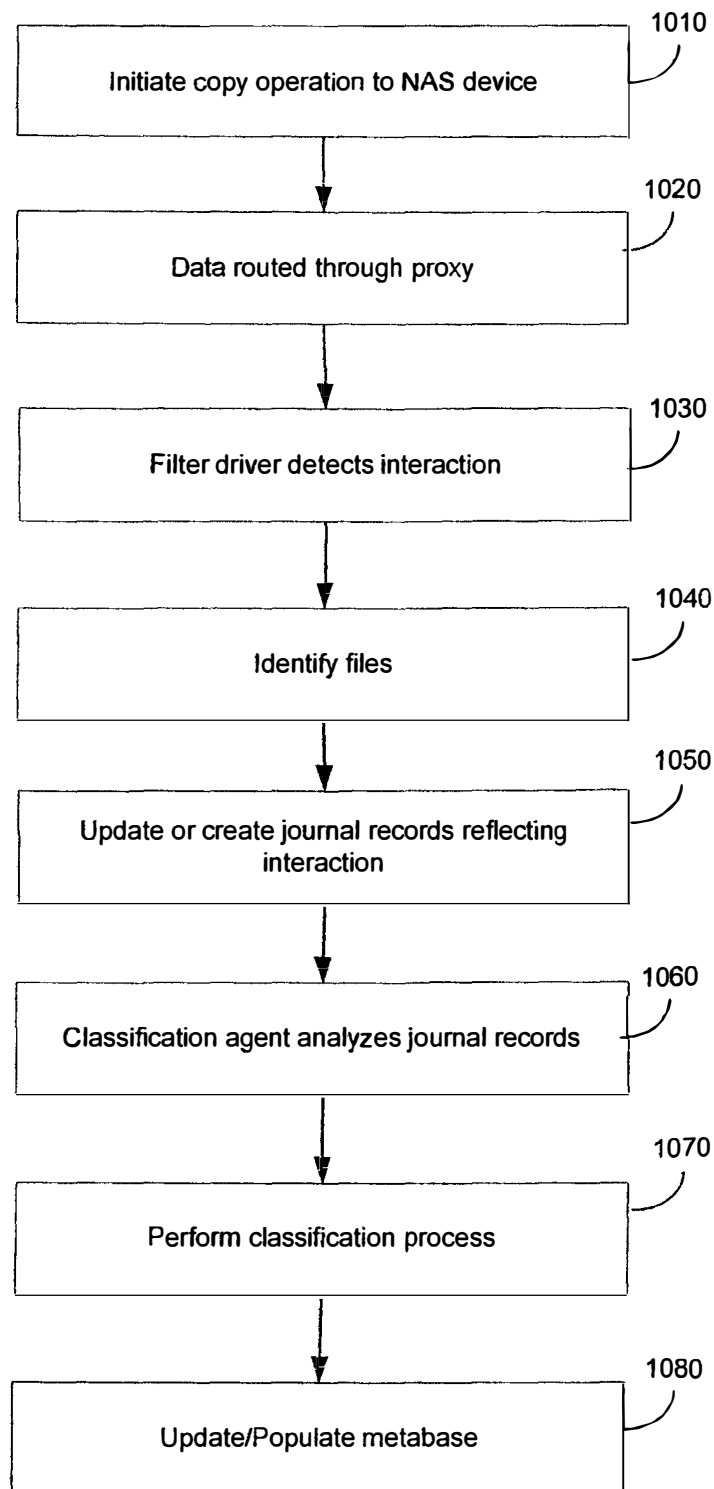


FIG. 10

U.S. Patent

May 25, 2010

Sheet 12 of 22

US 7,725,671 B2

1100

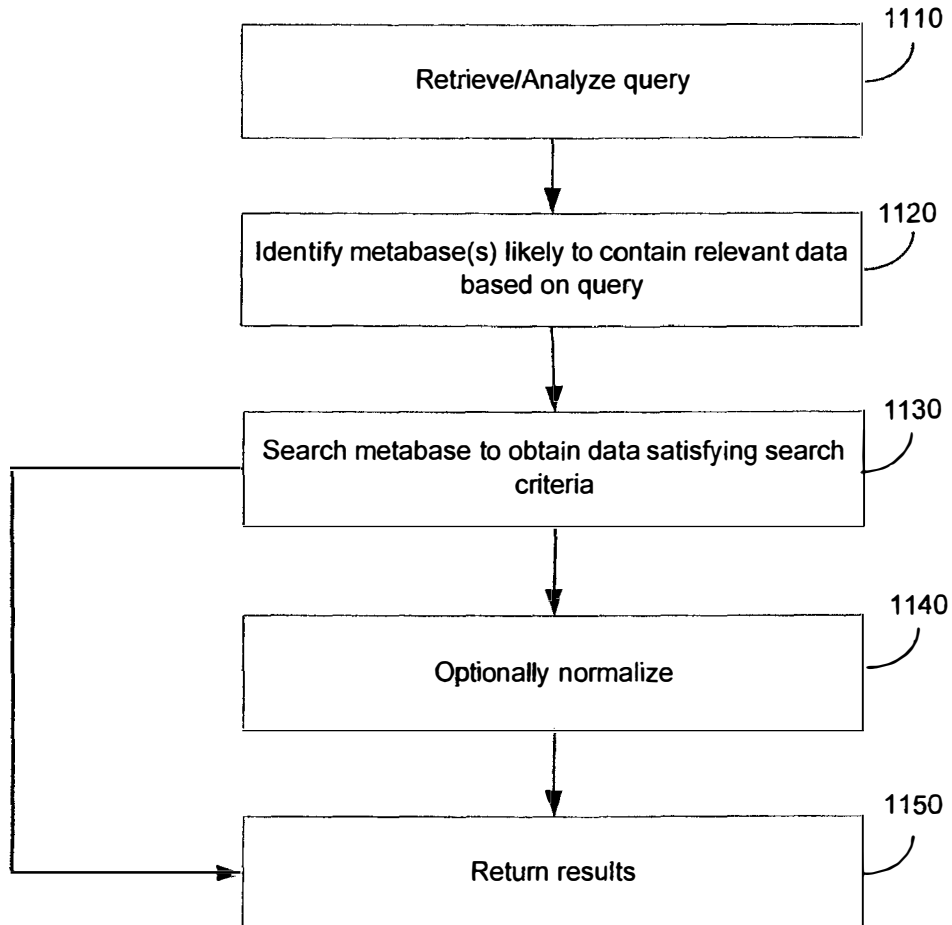


FIG. 11

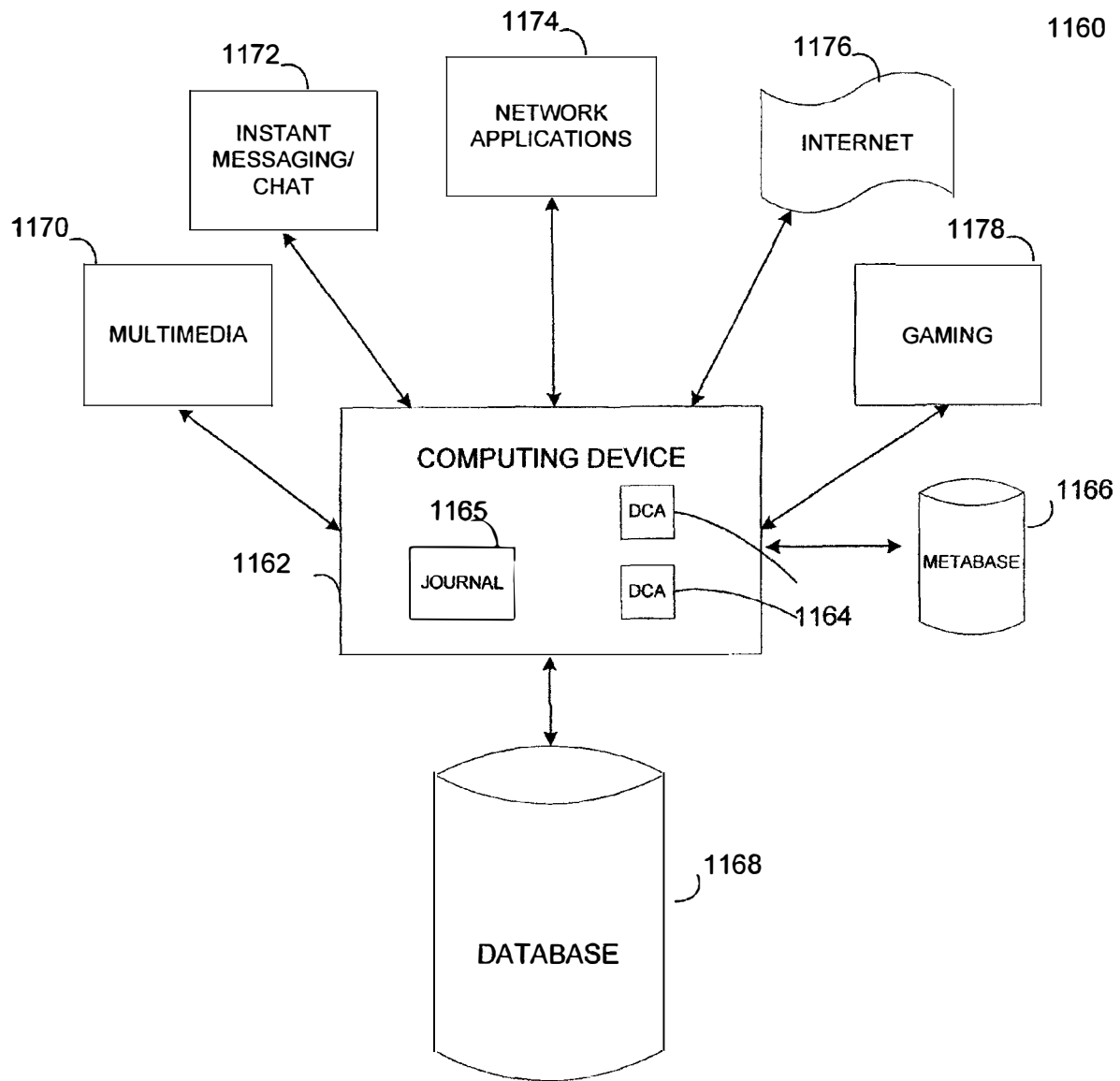


Figure 11a

1200

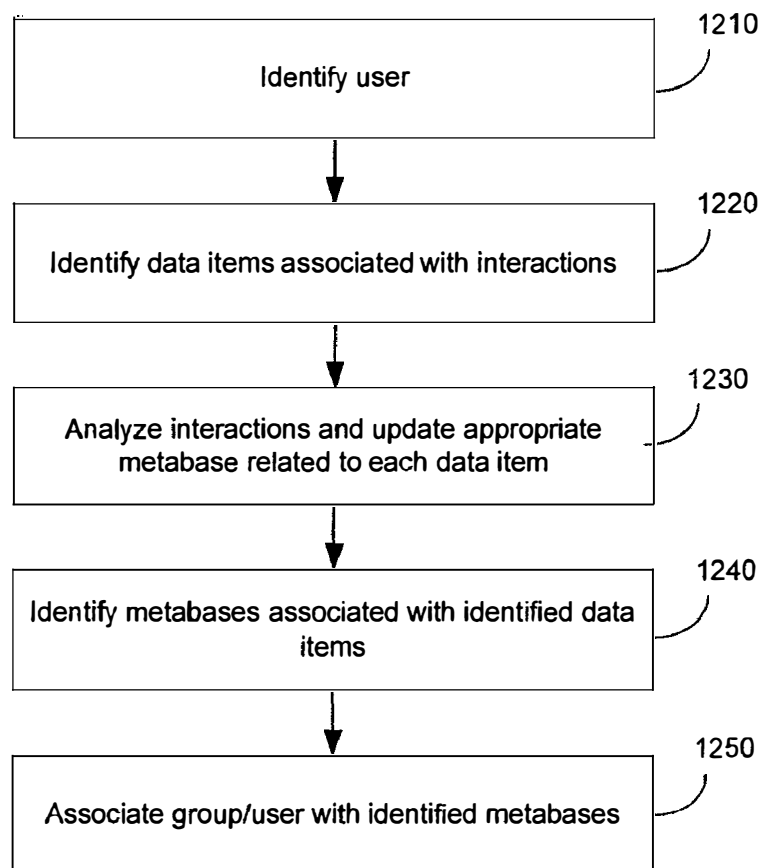


FIG. 12

1300

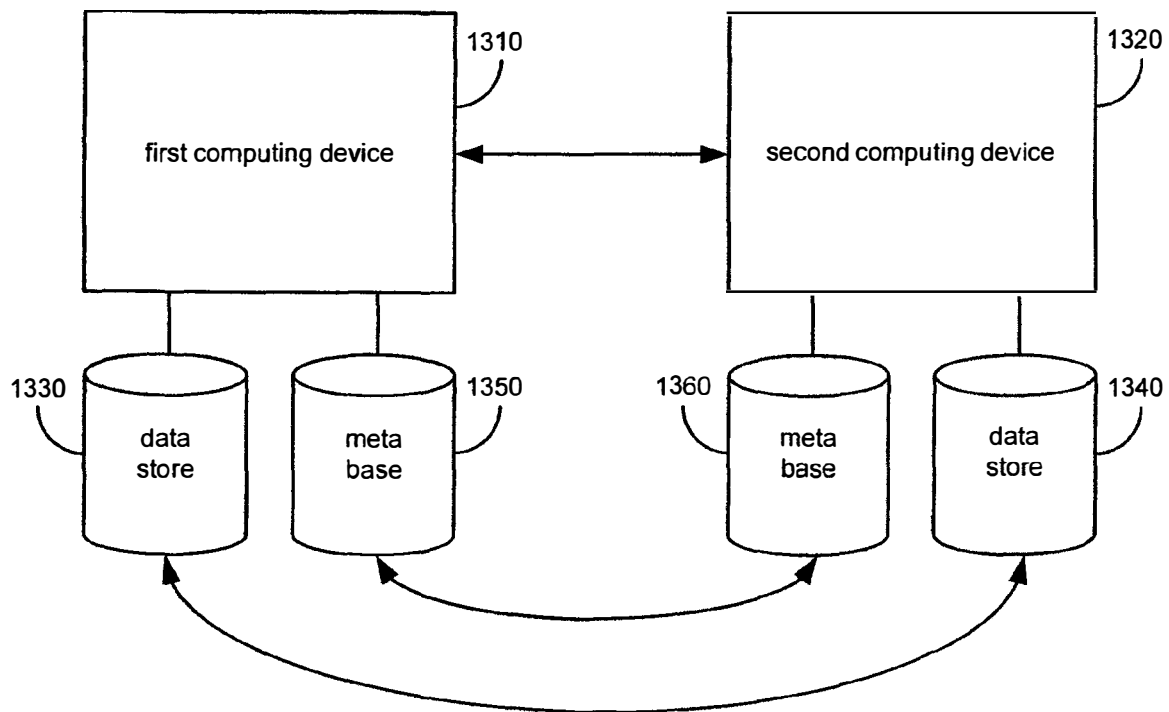


FIG. 13

1400

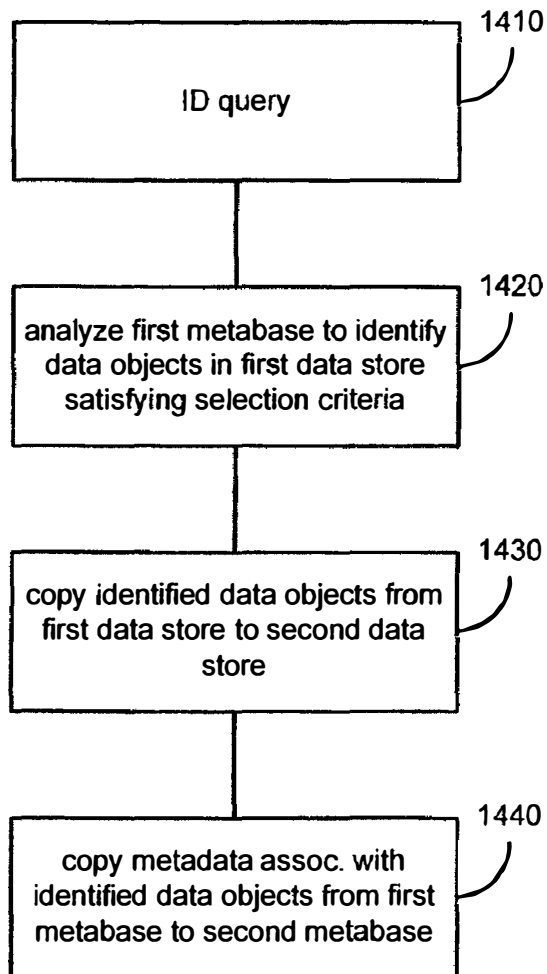


FIG. 14

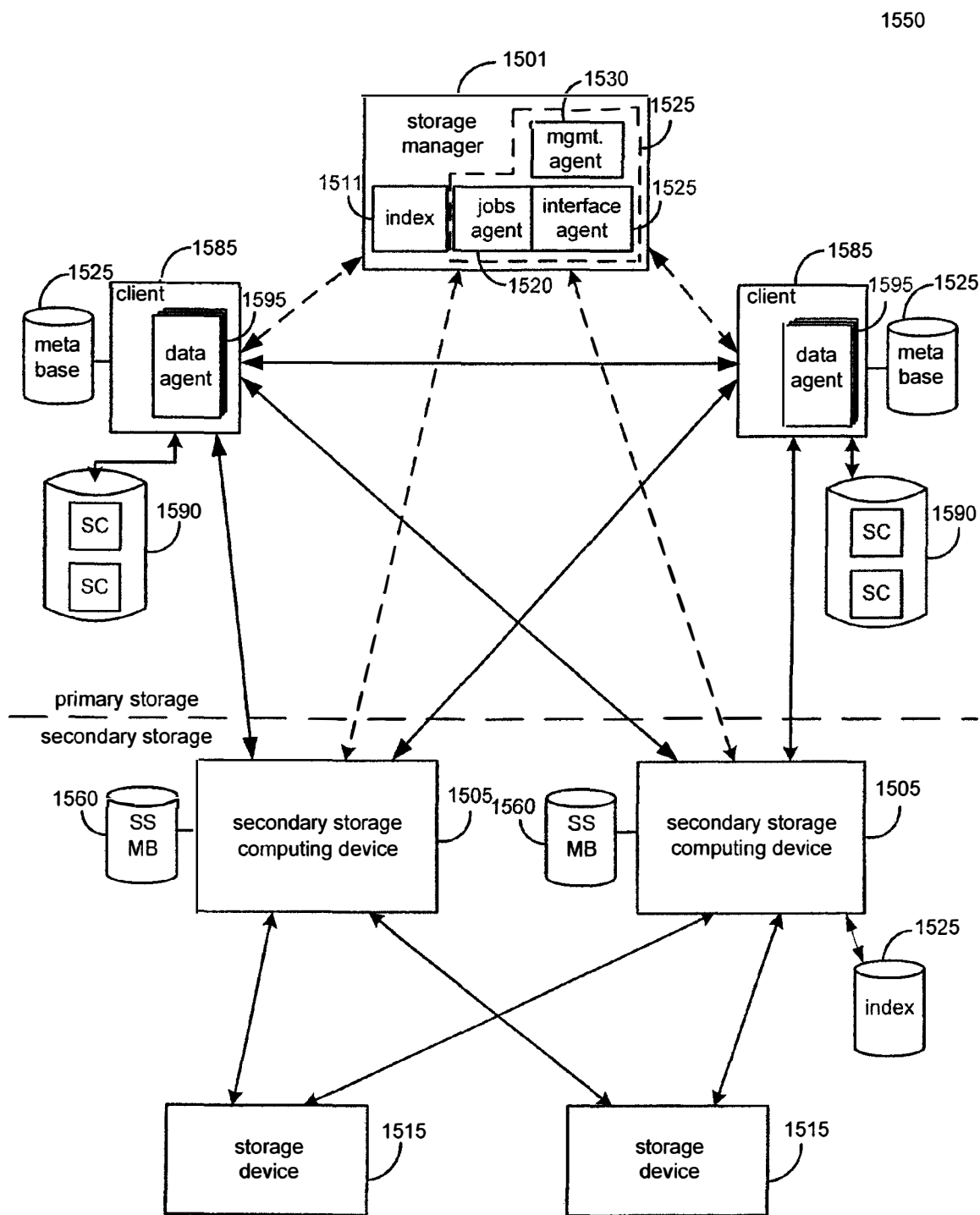


FIG. 15

U.S. Patent

May 25, 2010

Sheet 18 of 22

US 7,725,671 B2

1600

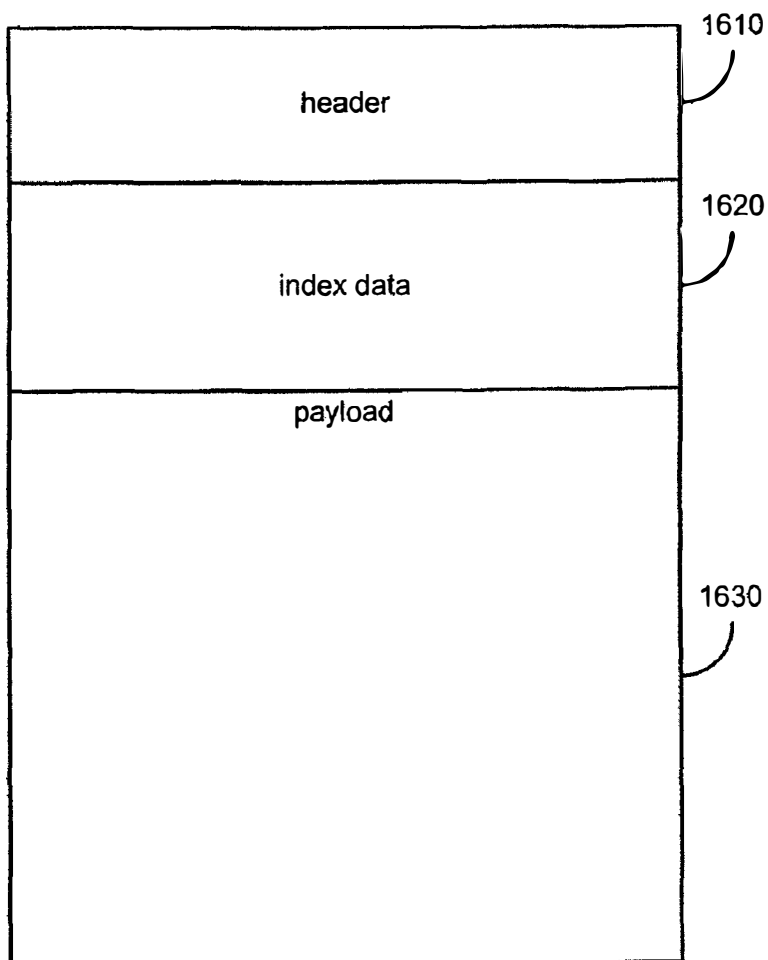


FIG. 16

U.S. Patent

May 25, 2010

Sheet 19 of 22

US 7,725,671 B2

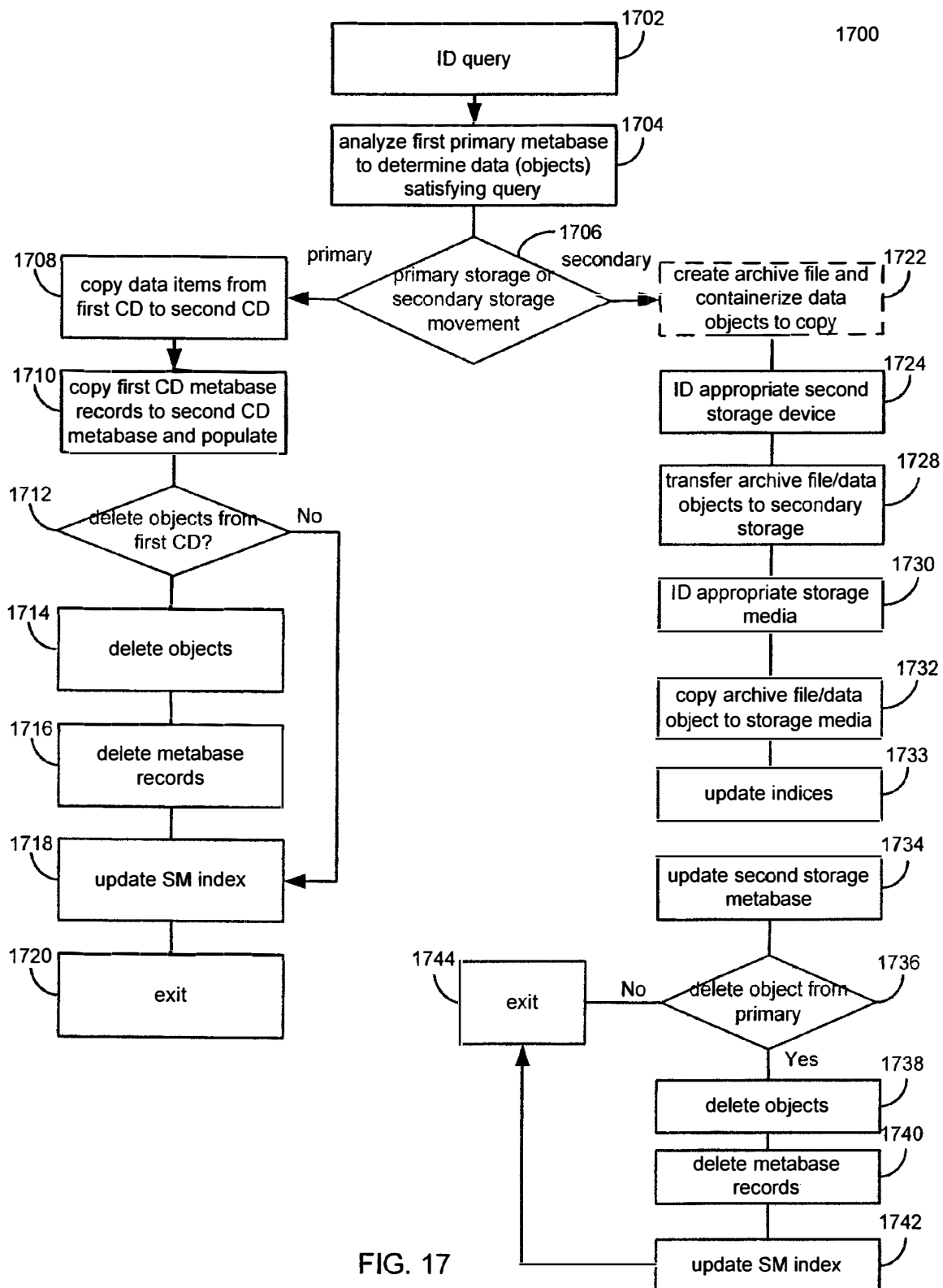


FIG. 17

1800

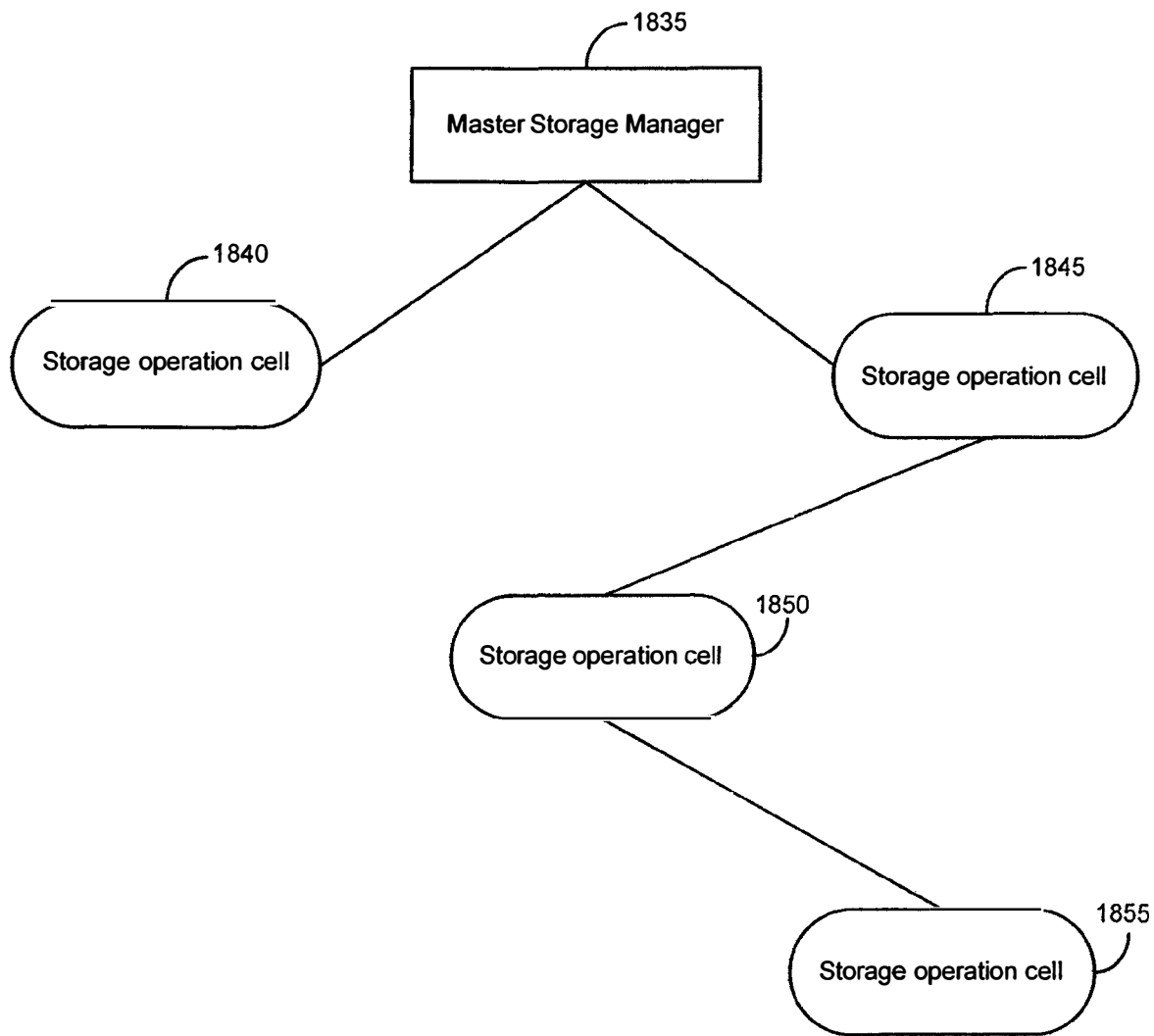


FIG. 18

U.S. Patent

May 25, 2010

Sheet 21 of 22

US 7,725,671 B2

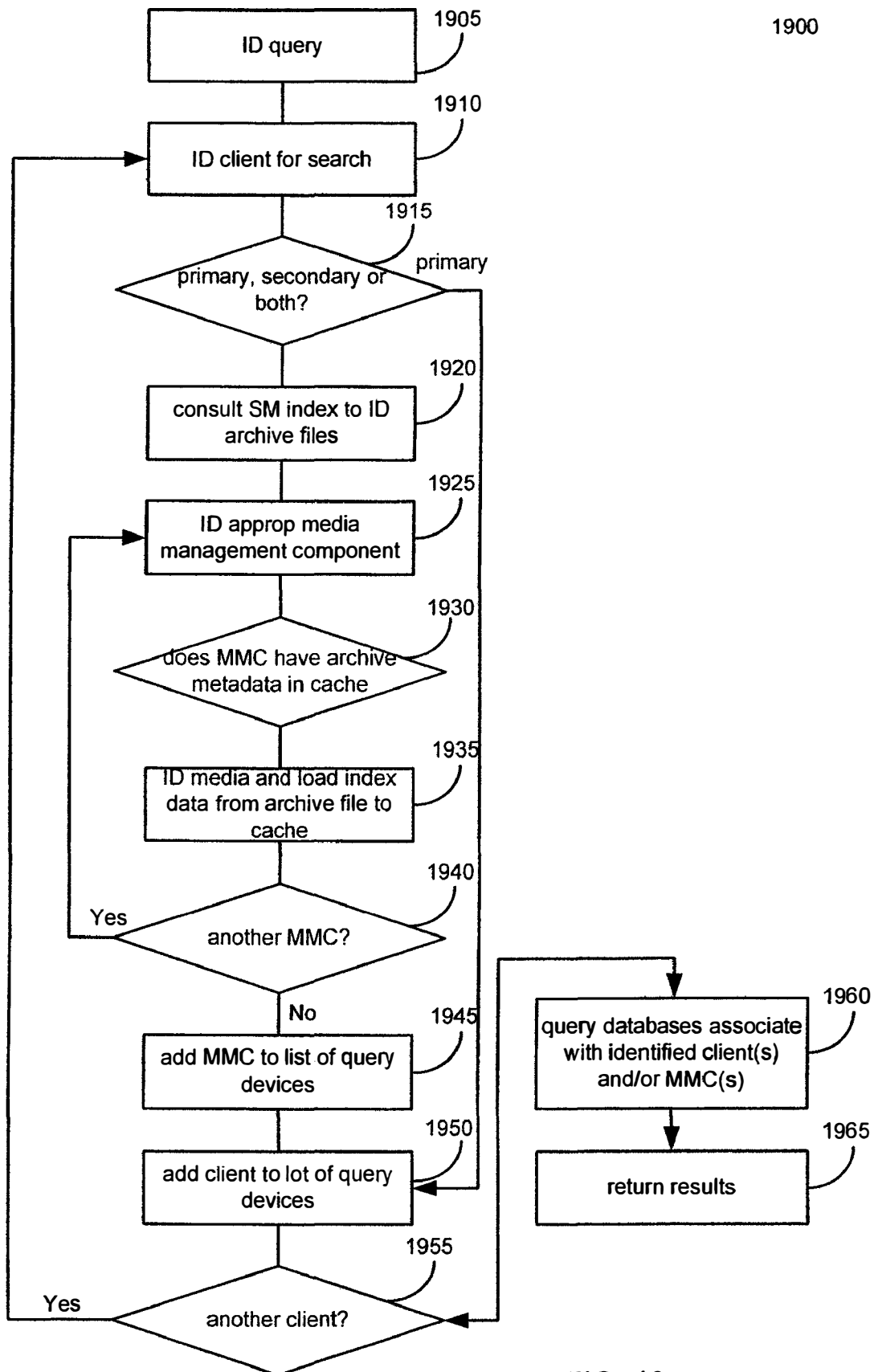


FIG. 19

U.S. Patent

May 25, 2010

Sheet 22 of 22

US 7,725,671 B2

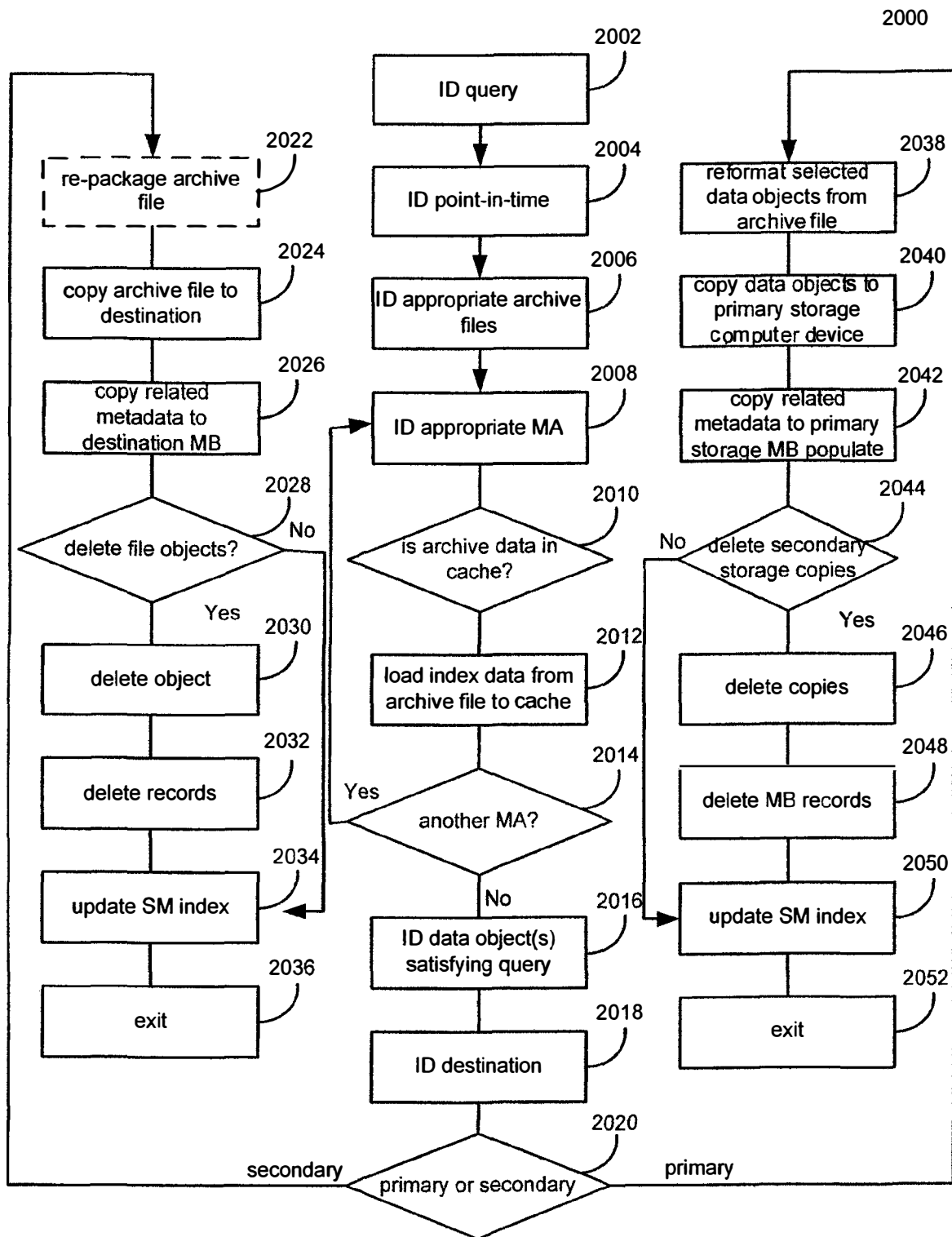


FIG. 20

US 7,725,671 B2

1

SYSTEM AND METHOD FOR PROVIDING REDUNDANT ACCESS TO METADATA OVER A NETWORK

CROSS-REFERENCE TO RELATED APPLICATION(S)

This application claims the benefit of U.S. Provisional Application No. 60/740,686, entitled SYSTEMS AND METHODS FOR CLASSIFYING INFORMATION IN A STORAGE NETWORK, filed Nov. 28, 2005, and U.S. Provisional Application No. 60/752,203, entitled SYSTEMS AND METHODS FOR CLASSIFYING AND TRANSFERRING INFORMATION IN A STORAGE NETWORK, filed Dec. 19, 2005, both of which are hereby incorporated herein by reference in their entirety.

This application is related to the following patents and pending U.S. applications, each of which is hereby incorporated herein by reference in its entirety:

application Ser. No. 09/354,058, titled HIERARCHICAL BACKUP AND RETRIEVAL SYSTEM, filed Jul. 15, 1999,

U.S. Pat. No. 6,418,478, titled PIPELINED HIGH SPEED DATA TRANSFER MECHANISM, issued Jul. 9, 2002, application Ser. No. 09/610,738, titled MODULAR BACKUP AND RETRIEVAL SYSTEM USED IN CONJUNCTION WITH A STORAGE AREA NETWORK, filed Jul. 6, 2000,

U.S. Pat. No. 6,542,972 titled LOGICAL VIEW AND ACCESS TO PHYSICAL STORAGE IN MODULAR DATA AND STORAGE MANAGEMENT SYSTEM, issued Apr. 1, 2003,

U.S. Pat. No. 6,658,436, titled LOGICAL VIEW AND ACCESS TO DATA MANAGE BY A MODULAR DATA AND STORAGE MANAGEMENT SYSTEM, issued Dec. 2, 2003,

application Ser. No. 10/658,095, titled DYNAMIC STORAGE DEVICE POOLING IN A COMPUTER SYSTEM, filed Sep. 9, 2003,

application Ser. No. 10/262,556, titled METHOD FOR MANAGING SNAPSHOTS GENERATED BY AN OPERATING SYSTEM OR OTHER APPLICATION, filed Sep. 30, 2002,

application Ser. No. 10/818,749, SYSTEM AND METHOD FOR DYNAMICALLY PERFORMING STORAGE OPERATIONS IN A COMPUTER NETWORK, filed Apr. 5, 2004,

application Ser. No. 10/877,831, HIERARCHICAL SYSTEM AND METHOD FOR PERFORMING STORAGE OPERATIONS IN A COMPUTER NETWORK, filed Jun. 25, 2004,

Application entitled SYSTEM AND METHOD FOR CONTAINERIZED DATA STORAGE AND TRACKING, filed Dec. 19, 2005,

application Ser. No. 60/752,198, entitled "Systems and Methods for Granular Resource Management in a Storage Network" filed Dec. 19, 2005,

application Ser. No. 11/313,224, entitled "Systems and Methods for Performing Multi-Path Storage Operations" filed Dec. 19, 2005,

Application Ser. No. 60/752,196, entitled "Systems and Methods for Migrating Components in a Hierarchical Storage Network" filed Dec. 19, 2005,

Application Ser. No. 60/752,202, entitled "Systems and Methods for Unified Reconstruction of Data in a Storage Network" filed Dec. 19, 2005,

2

Application Ser. No. 60/752,201, entitled "Systems and Methods for Resynchronizing Storage Operations" filed Dec. 19, 2005,

Application Ser. No. 60/752,197, entitled "Systems and Methods for Hierarchical Client Group Management" filed Dec. 19, 2005,

BACKGROUND

Aspects of the invention disclosed herein relate generally to performing operations on electronic data in a computer network. More particularly, aspects of the present invention relate to detecting data interactions within a computer network and/or performing storage-related operations for a computer network according to a specified classification paradigm.

Current storage management systems employ a number of different methods to perform storage operations on electronic data. For example, data can be stored in primary storage as a primary copy or in secondary storage as various types of secondary copies including, as a backup copy, a snapshot copy, a hierarchical storage management copy ("HSM"), as an archive copy, and as other types of copies.

A primary copy of data is generally a production copy or other "live" version of the data which is used by a software application and is generally in the native format of that application. Primary copy data may be maintained in a local memory or other high-speed storage device that allows for relatively fast data access if necessary. Such primary copy data is typically intended for short term retention (e.g., several hours or days) before some or all of the data is stored as one or more secondary copies, for example to prevent loss of data in the event a problem occurred with the data stored in primary storage.

Secondary copies include point-in-time data and are typically for intended for long-term retention (e.g., weeks, months or years depending on retention criteria, for example as specified in a storage policy as further described herein) before some or all of the data is moved to other storage or discarded. Secondary copies may be indexed so users can browse and restore the data at another point in time. After certain primary copy data is backed up, a pointer or other location indicia such as a stub may be placed in the primary copy to indicate the current location of that data.

One type of secondary copy is a backup copy. A backup copy is generally a point-in-time copy of the primary copy data stored in a backup format as opposed to in native application format. For example, a backup copy may be stored in a backup format that is optimized for compression and efficient long-term storage. Backup copies generally have relatively long retention periods and may be stored on media with slower retrieval times than other types of secondary copies and media. In some cases, backup copies may be stored at an offsite location.

Another form of secondary copy is a snapshot copy. From an end-user viewpoint, a snapshot may be thought of as an instant image of the primary copy data at a given point in time. A snapshot generally captures the directory structure of a primary copy volume at a particular moment in time, and also preserves file attributes and contents. In some embodiments, a snapshot may exist as a virtual file system, parallel to the actual file system. Users typically gain a read-only access to the record of files and directories of the snapshot. By electing to restore primary copy data from a snapshot taken at a given point in time, users may also return the current file system to the prior state of the file system that existed when the snapshot was taken.

US 7,725,671 B2

3

A snapshot may be created instantly, using a minimum of file space, but may still function as a conventional file system backup. A snapshot may not actually create another physical copy of all the data, but may simply create pointers that are able to map files and directories to specific disk blocks.

In some embodiments, once a snapshot has been taken, subsequent changes to the file system typically do not overwrite the blocks in use at the time of snapshot. Therefore, the initial snapshot may use only a small amount of disk space needed to record a mapping or other data structure representing or otherwise tracking the blocks that correspond to the current state of the file system. Additional disk space is usually only required when files and directories are actually modified later. Furthermore, when files are modified, typically only the pointers which map to blocks are copied, not the blocks themselves. In some embodiments, for example in the case of copy-on-write snapshots, when a block changes in primary storage, the block is copied to secondary storage before the block is overwritten in primary storage and the snapshot mapping of file system data is updated to reflect the changed block(s) at that particular point in time. An HSM copy is generally a copy of the primary copy data, but typically includes only a subset of the primary copy data that meets a certain criteria and is usually stored in a format other than the native application format. For example, an HSM copy might include only that data from the primary copy that is larger than a given size threshold or older than a given age threshold and that is stored in a backup format. Often, HSM data is removed from the primary copy, and a stub is stored in the primary copy to indicate its new location. When a user requests access to the HSM data that has been removed or migrated, systems use the stub to locate the data and often make recovery of the data appear transparent even though the HSM data may be stored at a location different from the remaining primary copy data.

An archive copy is generally similar to an HSM copy, however, the data satisfying criteria for removal from the primary copy is generally completely removed with no stub left in the primary copy to indicate the new location (i.e., where it has been moved to). Archive copies of data are generally stored in a backup format or other non-native application format. In addition, archive copies are generally retained for very long periods of time (e.g., years) and in some cases are never deleted. Such archive copies may be made and kept for extended periods in order to meet compliance regulations or for other permanent storage applications.

In some embodiments, application data over its lifetime moves from more expensive quick access storage to less expensive slower access storage. This process of moving data through these various tiers of storage is sometimes referred to as information lifecycle management ("ILM"). This is the process by which data is "aged" from more forms of secondary storage with faster access/restore times down through less expensive secondary storage with slower access/restore times, for example, as the data becomes less important or mission critical over time.

Examples of various types of data and copies of data are further described in the above-referenced related applications that are hereby incorporated by reference in their entirety. One example of a system that performs storage operations on electronic data that produce such copies is the QiNetix storage management system by CommVault Systems of Oceanport, N.J.

The QiNetix system leverages a modular storage management architecture that may include, among other things, storage manager components, client or data agent components, and media agent components as further described in U.S.

4

patent application Ser. No. 10/818,749 which is hereby incorporated herein by reference in its entirety. The QiNetix system also may be hierarchically configured into backup cells to store and retrieve backup copies of electronic data as further described in U.S. patent application Ser. No. 09/354,058 which is hereby incorporated by reference in its entirety.

Regardless of where data is stored, conventional storage management systems perform storage operations associated with electronic data based on location-specific criteria. For example, data generated by applications running on a particular client is typically copied according to location-specific criteria, such as from a certain location such as a specific folder or subfolder, according to a specified data path, etc. A module installed on the client or elsewhere in the system may supervise the transfer of data from the client to another location in a primary or secondary storage. Similar data transfers associated with location-specific criteria are performed when restoring data from secondary storage to primary storage. For example, to restore data, a user or system process must specify a particular secondary storage device, piece of media, archive file, etc. Thus, the precision with which conventional storage management systems perform storage operations on electronic data is generally limited by the ability to define or specify storage operations based on data location rather than information relating to or describing the data itself.

Moreover, when identifying data objects, such as files associated with performing storage operations, conventional storage systems often scan the file system of a client or other computing device to determine which data objects on the client should be associated with the storage operation. This may involve collecting file and/or folder attributes by traversing the file system of the client prior to performing storage operations. This process is typically time-consuming and uses significant client resources that might be more desirably spent performing other tasks associated with production applications. There is thus a need for systems and methods for performing more precise and efficient storage operations.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is illustrated in the figures of the accompanying drawings which are meant to be exemplary and not limiting, in which like references are intended to refer to like or corresponding parts, and in which:

FIG. 1 is a flow chart in accordance with an embodiment of the present invention;

FIG. 2 is a system constructed in accordance with an embodiment of the present invention;

FIG. 3a is a system constructed in accordance with an embodiment of the present invention;

FIG. 3b is a flow chart in accordance with an embodiment of the present invention;

FIG. 4 is a flow chart in accordance with an embodiment of the present invention;

FIG. 5 is a system constructed in accordance with an embodiment of the present invention;

FIG. 6 is a flow chart in accordance with an embodiment of the present invention;

FIG. 7 is a system constructed in accordance with an embodiment of the present invention;

FIG. 8 is a flow chart in accordance with an embodiment of the present invention;

FIG. 9 is a system constructed in accordance with an embodiment of the present invention;

FIG. 10 is a flow chart in accordance with an embodiment of the present invention;

US 7,725,671 B2

5

FIG. 11 is a flow chart in accordance with an embodiment of the present invention;

FIG. 11a is a system constructed in accordance with an embodiment of the present invention;

FIG. 12 is a flow chart in accordance with an embodiment of the present invention;

FIG. 13 is a system constructed in accordance with an embodiment of the present invention;

FIG. 14 is a flow chart in accordance with an embodiment of the present invention;

FIG. 15 is a system constructed in accordance with an embodiment of the present invention;

FIG. 16 is a data arrangement in accordance with an embodiment of the present invention;

FIG. 17 is a flow chart in accordance with an embodiment of the present invention;

FIG. 18 is a system constructed in accordance with an embodiment of the present invention;

FIG. 19 is a flow chart in accordance with an embodiment of the present invention; and

FIG. 20 is a flow chart in accordance with an embodiment of the present invention.

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosures, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

DETAILED DESCRIPTION

Aspects of the present invention are generally concerned with systems and methods that analyze and classify various forms of data that, among other things, facilitates identification, searching, storage and retrieval of data that satisfies certain criteria. Although described in connection with certain specific embodiments, it will be understood that the inventions disclosed herein have broad-based applicability to any wireless or hard-wired network or data transfer system that stores and conveys data from one point to another, including communication networks, enterprise networks, storage networks, and the like.

Aspects of the present invention provide systems and methods for data classification to facilitate and improve data management within an enterprise. The disclosed systems and methods evaluate and define data management operations based on data characteristics rather than data location, among other things. Also provided are methods for generating a data structure of metadata that describes system data and storage operations. This data structure may be consulted to determine changes in system data rather than scanning the data files themselves.

Generally, the systems and methods described in detail below are for analyzing data and other information in a computer network (sometimes referred to herein as a "data object") and creating a database or index of information which may describe certain pertinent aspects of the data objects that allow a user or system process to consult the database to obtain information regarding the network data. For example, a data collection agent may traverse a network file system and obtain certain characteristics and other attributes of data in the system. In some embodiments, such a database may be a collection of metadata and/or other infor-

6

mation regarding the network data and may be referred to herein as a "metabase." Generally, metadata refers to data or information about data, and may include, for example, data relating to storage operations or storage management, such as data locations, storage management components associated with data, storage devices used in performing storage operations, index data, data application type, or other data.

With this arrangement, if it is desired to obtain information regarding network data, a system administrator or system process may simply consult the metabase for such information rather than iteratively access and analyze each data item in the network. Thus, this significantly reduces the amount of time required to obtain data object information by substantially eliminating the need to obtain information from the source data, and furthermore minimizes the involvement of network resources in this process, substantially reducing the processing burden on the host system.

Various embodiments of the invention will now be described. The following description provides specific details for a thorough understanding and enabling description of these embodiments. One skilled in the art will understand, however, that the invention may be practiced without many of these details. Additionally, some well-known structures or functions may not be shown or described in detail, so as to avoid unnecessarily obscuring the relevant description of the various embodiments.

The terminology used in the description presented below is intended to be interpreted in its broadest reasonable manner, even though it is being used in conjunction with a detailed description of certain specific embodiments of the invention. Certain terms may even be emphasized below; however, any terminology intended to be interpreted in any restricted manner will be overtly and specifically defined as such in this Detailed Description section.

A flow chart 100 illustrating some of the steps associated with one embodiment of the present invention is shown in FIG. 1. In order to perform some of the functions described herein, it may be necessary at the outset to install certain data classification software or data classification agents on computing devices within at least parts of the network (step 102). This may be done, for example, by installing classification software on client computers and/or servers within a given network. In some embodiments, classification agents may be installed globally on a computing device or with respect to certain subsystems on a computing device. The classification software may monitor certain information regarding data objects generated by the computers and classify this information for use as further described herein.

Next, at step 104, a monitor agent may be initialized. Such a monitoring agent may be resident or installed on each computing device similar to the deployment of classification agents described above and may be configured to monitor and record certain data interactions within each machine or process. For example, the monitor agent may include a filter driver program and may be deployed on an input/output port or data stack and operate in conjunction with a file management program to record interactions with computing device data. This may involve creating a data structure such as a record or journal of each interaction. The records may be stored in a journal data structure and may chronicle data interactions on an interaction by interaction basis. The journal may include information regarding the type of interaction that has occurred along with certain relevant properties of the data involved in the interaction. One example of such a monitor program may include Microsoft's Change Journal or similar program.

US 7,725,671 B2

7

Prior to populating a metabase with metadata, the portions of the network or subject system may be quiesced such that no data interactions are permitted prior to completing an optional scan of system files as described in conjunction with step 106 below. This may be done in order to obtain an accurate point in time picture of the data being scanned and to maintain referential integrity within the system. For example, if the system were not quiesced, data interactions would continue and be allowed to pass through to mass storage and thus data would change. However, in some embodiments, the subject system may be allowed to continue to operate, with operations or instructions queued in a cache. These operations are typically performed after the scan is complete so that any such data interactions occur based on the cached operations are captured by the monitor agent.

Generally, the file scanning of step 106 may be performed by a data classification agent and may include traversing the file system of a client to identify data objects or other files, email or other information currently stored or present in the system and obtain certain information regarding the information such as any available metadata. Such metadata may include information about data objects or characteristics associated with data objects such as the data owner (e.g., the client or user that generates the data or other data manager), the last modified time (e.g., the time of the most recent modification), the data size (e.g., number of bytes of data), information about the data content (e.g., the application that generated the data, the user that generated the data, etc.), to/from information for email (e.g., an email sender, recipient or individual or group on an email distribution list), creation date (e.g., the date on which the data object was created), file type (e.g., format or application type), last accessed time (e.g., the time the data object was most recently accessed or viewed), application type (e.g., the application which generated the data object), location/network (e.g., a current, past or future location of the data object and network pathways to/from the data object), frequency of change (e.g., a period in which the data object is modified), business unit (e.g., a group or department that generates, manages or is otherwise associated with the data object), and aging information (e.g., a schedule, which may include a time period, in which the data object is migrated to secondary or long term storage), etc. The information obtained in the scanning process may be used to initially populate the metabase of information regarding network data at step 108.

After the metabase has been populated, the network or subject system may be released from the quiesced state and normal operation may resume. Next, at step 110, the monitor agent may monitor system operations to record changes to system data in the change journal database as described above. The change journal database may include a database of metadata or data changes and may comprise log files of the data or metadata changes. In some embodiments, the data classification agent may periodically consult the change journal database for new entries. If new entries exist, these entries may be examined, and if deemed relevant, then analyzed, parsed, and written to the metabase as an update (step 112). In other embodiments, change journal entries may be supplied substantially in parallel to the journal database and data classification agent. This allows the metabase to maintain substantially current information regarding the state of system data at any given point in time.

As mentioned above, one benefit of such a metabase is it significantly reduces the amount of time required to obtain information by substantially eliminating the need to obtain information directly from the source data. For example, assume a system administrator desires to identify data objects

8

that a certain user has interacted with that contain certain content or other characteristics. Rather than search each file in each directory, a very time consuming process, the administrator may simply search the metabase to identify such data objects and any properties associated with those objects, (e.g., metadata, which may include location, size, etc.), resulting in significant time savings.

Moreover, use of the metabase for satisfying data queries also minimizes the involvement of network resources in this process, substantially reducing any processing burden on the host system. For example, as described above, if an administrator desires to identify certain data objects, querying the metabase rather than the file system virtually removes the host system from the query process (i.e., no brute force scanning of directories and files required), allowing host computing devices to continue performing host tasks rather than be occupied with search tasks.

FIG. 2 shows one embodiment of a client 200 constructed in accordance with principles of the present invention. As shown, client 200 may include a classification agent 202 and a monitor agent 206, which, in some embodiments, may be combined as an update agent 204, and which may be a single module encompassing the functionality of both agents. Client 200 may also include an internal or external data store 209, metabase 210, and change record 212.

Generally, client 200 may be a computing device, or any portion of a computing device that generates electronic data. Data store 209 generally represents application data such as production volume data used by client 200. Metabase 210, which may be internal or external to client 200 may contain information generated by classification agent 202 as described above. Similarly, change journal 212, which also may be internal or external to client 200, may contain information generated by monitor agent 206 as described above.

In operation, data interactions occurring within client 200 may be monitored with update agent 204 or monitor agent 206. Any relevant interaction may be recorded and written to change record 206. Data classification agent 202 may scan or receive entries from monitor agent 206 and update metabase 210 accordingly. Moreover, in the case where update agent 204 is present, monitored data interactions may be processed in parallel with updates to change record 212 and written to data store 208 and metabase 210 occurring accordingly. A file system 207 may be used to conduct or process data from the client to a data store 209.

FIG. 3a provides a more detailed view of the journaling and classification mechanisms of client 200 generally shown in FIG. 2. As shown, system 300 may include a memory 302, an update agent 304 which may include a separate or integrated monitor agent 306, classification agents 312a and 312b, a content agent 315, a monitor program index 310, metabase 314 and mass storage device 318.

In operation, data interactions that occur between memory 302 and mass storage device 318 may be monitored by monitor agent 306. In some embodiments, memory 302 may include random access memory (RAM) or other memory device employed by a computer in client 200 in performing data processing tasks. Certain information from memory 302 may be periodically read or written to mass storage device 318 which may include a magnetic or optical disk drive such as a hard drive or other storage device known in the art. Such data interactions are monitored by monitoring agent 306 which, in some embodiments, may include any suitable monitoring or journaling agent as further described herein.

As shown, system 300 may also include an administrative file system program 316, such as a file system program known in the art, which may include operating system programs, a

US 7,725,671 B2

9

FAT, an NTFS, etc. that may be used to manage data movement to and from mass storage device 318. Thus, in operation, data may be written from memory 302 to mass storage device 318 via file system program 316. Such an operation may occur, for example, to access data used to service an application running on a computing device. During this operation, monitor agent 306 may capture this interaction and generate a record indicating that an interaction has occurred and store the record in index 310. The data may be stored in mass storage 318 under the supervision of file system manager 316.

As shown in FIG. 3a, monitor agent 306 may analyze data interactions such as interactions between memory 302 and mass storage 318 via the file system manager 316, and record any such interactions in monitor index 310. Thus, as described above, monitor index 310 may represent a list of data interactions wherein each entry represents a change that has occurred to client data along with certain information regarding the interaction. In embodiments where Microsoft Change Journal or other similar software is used, such entries may include a unique identifier such as an update sequence number (USN), certain change journal reason codes identifying information associated with a reason(s) for the change made, along with data or metadata describing the data and certain data properties, data copy types, etc.

Thus, in operation, as data moves from memory 302 to mass storage 318 (or vice versa), monitor agent 304 may create and write an entry to index 310 which may in turn, be analyzed and classified by classification agent 312b for entry in metabase 314. In some embodiments, classification agent 312a may be coupled with mass storage device (either directly or through file system manager 316) and write metadata entries to both metabase 314 and mass storage device 318. In some embodiments, the metabase information may be stored on mass storage device 318. Moreover, in an alternate embodiment, classification agent 312b may periodically copy or backup metabase 314 to the storage device under the direction of a storage manager and/or pursuant to a storage policy (not shown) such that the information in metabase 314 may be quickly restored if lost, deleted or is otherwise unavailable.

In some embodiments, optional classification agent 312a may operate in conjunction with monitor agent 306 such that data moving to mass storage device 318 is classified as further described herein and written to device 318. With this arrangement, the data, along with the processed metadata describing that data, is written to mass storage device 318. This may occur, for example in embodiments in which monitor agent 306 and classification agent 312a are combined into update agent 304. Writing metadata in this way allows it to be recovered or accessed from mass storage device 318 if necessary, for example, when metabase 314 is missing certain information, busy, or otherwise inaccessible.

Content agent 315 may be generally used to obtain or filter data relating to content of the data moving from memory 302 to mass storage 318. For example, content agent 315 may read data payload information and generate metadata based on the operation for storage in metabase 314 and may include a pointer to the data item in mass storage 318. The pointer information may optionally be stored in an index. This metadata may also be stored with the data item in mass storage 318 or as an entry functioning in place of or in addition to metabase 314. Storing metadata relating to data content in metabase 314 provides the capability to perform content searches for data in the metabase 314, instead of searching entries in mass storage 318. This allows the system to quickly locate information satisfying a content query that may be retrieved from mass storage 318, if necessary.

10

Moreover, such content metadata may be generated and used in locating data based on content features throughout a hierarchy within a storage operation system (e.g., content metadata may be generated and stored at each or certain levels of storage within the system (primary secondary, tertiary, etc.) to facilitate location and retrieval of data based on content). As will be understood by one of skill in the art, the functionality provided by the content agent 315, classification agent 312a & b and monitor agent 306 may be provided by one or more modules or components such that the modules may be integrated into a single module providing the described functions, or may be implemented in one more separate modules each providing some of the functions.

FIG. 3b is a flow chart 350 illustrating some of the steps that may be involved in the journaling process described above. At step 355, the monitor program may be initialized, which may include instantiating a data structure or index for recording interaction entries, and the assignment of a unique journal ID number which allows the system to differentiate between various journaling data structures that may be operating within the system. As mentioned above, the monitor program may include a filter driver or other application that monitors data operations (step 360). During the monitoring process, the monitor agent may observe data interactions between memory and mass storage to determine that certain data interactions have occurred. Information relating to the interactions may be captured and used to populate the metabase. In some instances, interaction types or certain aspects of interactions are captured. Such types or aspects may be defined in an interaction definition, which may be a change journal reason codes as used by Microsoft's Change Journal program, or be defined by a user or network administrator to capture some or all data interactions in order to meet user needs. For example, certain change definitions may record every data interaction that occurs regardless of whether any data actually changes or not. Such information may be useful, for example, to determine users or processes that have "touched scanned or otherwise accessed data without actually changing it.

Thus, it is possible to employ interaction definitions that may capture a relatively broad or narrow set of operations, allowing a user to customize the monitor program to meet certain goals. Such interaction definitions may define or describe data movement, changes, manipulations or other operations or interactions that may be of interest to a system user or administrator (e.g., any operation that "touches" data may be recording along with the action or operation that caused the interaction (e.g. read, write, copy, parse, etc.) Moreover, change definitions may evolve over time or may be dynamic based on the entries sent to the index. For example, if expected results are not obtained, change definitions may be modified or additional definitions used until appropriate or desired results are obtained. This may be accomplished, for example by globally linking certain libraries of interaction definitions and selectively enabling libraries on a rolling basis until acceptable results are achieved. This process may be performed after the initial activation of the monitor agent and periodically thereafter, depending on changing needs or objectives.

Moreover, in some embodiments, the system may support the use of "user tags" that allow certain types of information to be tagged so they may be identified and tracked throughout the system. For example, a user may designate a particular type of data or information such as project information, or information shared between or accessed by particular group of users to be tracked across the system or through various levels of storage. This may be accomplished through a user

US 7,725,671 B2

11

interface (not shown) that allows a user to define certain information to be tagged, for example, by using any available attribute within the system such as those specified above with respect to the classification agent or filter used in the system. In some embodiments, the user may define one or more tags using these or other attributes which may be further refined by combining them through the use of logical or Boolean operators to a define a certain tag expression.

For example, a user may define a certain tag by specifying one or more criteria to be satisfied such as certain system users, a certain data permission level, a certain project, etc. These criteria may be defined using a logical operators such as an AND or OR operators to conditionally combine various attributes to create a condition that defines a tag. All information satisfying those criteria may be tagged and tracked within the system. For example, as data passes through the monitor agent 306 (or other module within update agent 304), the data satisfying these criteria may be identified and tagged with a header or a flag or other identifying information as is known in the art. This information may be copied or otherwise noted by metabase 314 and mass storage 318 so that the information may be quickly identified. For example, the metabase may contain entries keeping track of all entries satisfying the tag criteria along with information relating to the types of operations performed on the information as well as certain metadata relating to the data content and its location in mass storage 318. This allows the system to search the metabase at a particular level of storage for the information, and quickly locate it within mass storage device for potential retrieval.

Next, a step 365, the monitor agent may continue to monitor data interactions based on change definitions until an interaction satisfying a definition occurs. Thus, a system according to one embodiment of the present invention may continue to monitor data interactions at steps 360 and 365 until a defined interaction, such as an interaction satisfying or corresponding to a selection criterion, such as an interaction definition, etc., occurs. If a defined interaction does occur, the monitor agent may create a record, which may be stored in a monitor program index, and in some embodiments, assign an interaction code that describes the interaction observed on the data object. Next, a step 370, the monitor program may identify a data object identifier associated with the data and that is associated with the data interaction, such as a file reference number (FRN) related to the data object. The FRN may include certain information such as the location or path of the associated data object. Any additional information (e.g., data properties, copy properties, storage policy information, etc.) associated with the FRN may also be obtained in order to enrich or enhance the record. In some embodiments, this may further involve obtaining information from other system files including master file tables (MFTs) to further enhance the metabase entries. Additional processing or formatting of the metabase entries may also occur in accordance with certain defined classification paradigms in order to populate the metabase with optimal or preferred information.

Next, at step 375 the record may be assigned record identifier such as a unique update sequence number (USN) that may be used to uniquely identify the entry within the index, and, in some embodiments, act as an index memory location. Thus a particular record may be quickly located with a data structure based on the USN. Next at step 380, the information described above may be concatenated or otherwise combined with other data or metadata data obtained by the monitor agent and arranged in an appropriate or expected format to create the record that may be written to the monitor index.

In alternate embodiments, the information described above may be written to the index and arranged at the index into an

12

expected format or may be written to the record "as received" and include metadata or other information, such as a header describing that information such that adherence to a strict data format is not required. For example, some records may contain more or less information than other records, as appropriate. After the record has been constructed and deemed complete, the record may be "closed" by the system at step 385 and the system may then assign another USN to the next detected change. However, if the record is determined to be incomplete, the monitor agent or update agent may request any missing information to complete the entry. If such information is not received, the monitor agent may set a flag within the record to indicate it contains incomplete information and the record may then be closed.

FIG. 4 is a flow chart 400 illustrating some of the steps that may be involved in a data classification process. At step 410, the classification agent may be initialized, which may include activating, reserving and/or clearing certain buffers and/or linking to libraries associated with deployment of the classification agent. Prior to scanning the interaction records generated by the monitor agent as described above, the classification agent may classify existing stored data by, for example, traversing the file and directory structure of an object system to initially populate the metabase as described herein.

Next, at step 420, during normal operation, the classification agent may scan the entries in the interaction journal to determine whether new entries exist since any previous classification processing was performed, for example, by determining whether the most recent entry currently in the journal is more or less recent than the last journal entry analyzed. This may be accomplished in several ways. One method may include scanning a time or date information associated with the last journal entry examined and comparing it to the most recent time or date information than the entry currently present in the journal. If it is determined that the most recent journal entry occurred after a previous classification process, this process may be performed iteratively by "walking backwards" through the journal entries until the last entry previously analyzed by the classification agent is found. All entries with time information after that point may be considered new or unprocessed by the classification agent (step 440). If the last entry analyzed has the same time stamp as the most recent entry in the journal, no new entries exist and the system may return to step 420 to continue monitoring, etc.

Another method of identifying new journal entries may include comparing record identifiers such as USN numbers assigned to each journal entry (step 430). Journal entries with a larger USN number than the last entry previously analyzed may be considered new or unprocessed. If the last entry analyzed has the same USN number as the current entry, no new entries exist and the system may return to step 420 to continue monitoring, etc. This comparison may be performed until new entries are located (step 440) or until it is determined that no new entries exist.

In other embodiments, rather than scanning the journal data structure for new entries, any entries created by the journal agent may be automatically sent to the classification agent and the identification process may be largely unnecessary (except for the case where such scanning is necessary or desirable, for example, to repopulate the journal or verify certain information, etc.).

Next, at step 450, assuming new journal entries are identified, the system may determine if a metabase record already exists for the data object associated with those entries. This may be accomplished by comparing data object identifiers, such as FRNs of metabase entries with data object identifiers

US 7,725,671 B2

13

such as FRNs of journal entries. Matching these and other unique data characteristics may be used to match or correlate metabase and journal entries.

If no corresponding metabase record is identified, a new record may be created at step 460. This may involve creating a new metabase entry ID, analyzing the journal entry and parsing the entry into a predetermined format, and copying certain portions of the parsed data to the new metabase entry (steps 460 and 470), as further described herein. Any additional metadata or file system information may also be associated with the new entry to enhance its content, including information from an FRN or information derived from an interaction code present in the journal entry, file system, e.g., MFT, etc. (step 480).

On the other hand, if a corresponding metadata entry is identified, the new journal entry may be processed as described above and may overwrite some or all of the corresponding entry. Such an updated pre-existing entry may receive an updated time stamp to indicate a current revision. However, in some embodiments, even if a corresponding entry is located, a new entry may be created and written to the metabase and optionally associated with the existing record. In such a system, the older related record may be maintained, for example, archival, recreation, historical or diagnostic purposes, and in some embodiments, may be marked or indicated as outdated or otherwise superseded. Such corresponding entries may be linked to one another via a pointer or other mechanism such that entries relating to the history of a particular data object may be quickly obtained.

Next, at step 490 the system may process any additional new journal entries detected by returning to step 450, where those additional new entries may be processed as described above. If no new entries are detected, the system may return to step 420 to perform additional scans on the journal data structure and continue monitoring.

FIG. 5 illustrates an embodiment of the present invention in which a secondary processor performs some or all of functions associated with the data classification process described herein, including certain search functions. As shown, system 500 may include a manager module 505 which may include an index 510, a first computing device 515, (which may include a first processor 520, a journal agent 530, and a data classification agent 535), and a second computing device 540 which may include a second processor 545 and a data classification agent 535. System 500 may also include data store 550, a metabase 555 and change journal 560.

Computing devices 515 and 544 may be any suitable computing device as described herein and may include clients, servers or other network computers running software, such as applications or programs that create, transfer, and store electronic data. In some embodiments, metabase 555 and journal 560 may be physically located within computing device 515, e.g., stored on local mass storage. In other embodiments the metabase 555 and journal 560 may be external to computing device 515 (or distributed between the two). In yet other embodiments, metabase 555 is accessible via a network and journal 560 is a local device.

In operation, computing device 515 may operate in a substantially similar manner as system 300 shown in FIG. 3 with second processor 545 in second computing device 540 performing certain functions. For example, as shown, data classification agent 535 and journaling agent 530 may operate substantially as described in connection with FIG. 3, i.e., journaling agent monitors data interactions on computing device 515 and records the interactions in journal 535 and classification agent processes journal entries and populates metabase 555, etc.

14

However, certain of the functions may be initiated or performed in whole or in part by second processor 545. Computing operations associated with journal agent 530 and/or classification agent 535 may run on or be directed by second processor 545 and may also utilize support resources located on or associated with computing device 540 such that the resources on computing device 515 are substantially unimpacted by these operations. This may serve to offload certain non-critical tasks from the host system (515) and have them performed by a secondary computing device 545.

For example, in some embodiments, the processing burden associated with some or all of the following tasks normally performed by first computing device 515 may be performed by processor 545 and associated resources in second computing device 540: (1) the initial scan of client files by the classification agent 535 and population of metabase 555, (2) the ongoing monitoring of data interactions of computing device (e.g., 515) and generation of interaction records for storage in journal 560, (3) processing and classification of journal information for updating metabase 555; and (4) searching or otherwise analyzing or accessing metabase 555 and/or journal 560 for certain information. However, in some embodiments it may be preferred to assign the secondary computing device the certain tasks such as those associated with searching metabase 555, while other tasks such as updating the journal and metabase may be performed by the primary computing device.

Performing such operations using a secondary or other processor may be desirable, for example, when the primary processor (e.g., processor 520) is unavailable, over utilized, unavailable or otherwise heavily used, or when it is otherwise desired to remove the primary processor and other primary system resources from performing certain tasks such as the ones described above. For example, in the case where it is desired to search or access metabase 555 for certain information, it may be preferable to avoid using processor 520 for this task so it remains free to perform other tasks associated with programs operating on computing device 515 (e.g., when computing device 515 is busy performing other network or application-related functions).

In some embodiments, the secondary processor may be located on computing device 515 (e.g., processor 525) and may perform the operations described herein in connection with processor 545. Moreover, some embodiments may include a manager module 505 which may coordinate overall operations between the various computing devices. For example, manager module 505 may monitor or otherwise be cognizant of the processing load on each computing device and may assign processing tasks based on availability (e.g., load balance). For example, if processor 520 is idle or operating at a low capacity, a request to search metabase 555 may be handled by processor 520. However, if processor 520 is busy or scheduled to perform or is performing priority work, manager 505 may assign the task to processor 545. Manager 505 may act as a general arbiter for such processor assignments to ensure system 500 is making efficient use of system resources.

FIG. 6 is flow chart 600 illustrating some of the steps involved in performing a query on a metabase in a multiple processor system similar to the system shown in FIG. 5. At step 610, a query may be received by the system for certain information. This request may be processed and analyzed by a manager module or other system process that determines or otherwise identifies which metabase or metabases within the system likely include at least some of the requested information, step 630. For example, the query itself may suggest which metabases to search and/or the management module

US 7,725,671 B2

15

may consult an index that contains information regarding metabase content within the system as further described herein. It will be understood that the identification process may require searching and identifying multiple computing devices within an enterprise or network that may contain information satisfying search criteria.

In other embodiments, search requests may be automatically referred to a secondary processor to minimize processing demands on the computing device that may have created or is otherwise associated with the identified metabase(s). In some embodiments, it is preferable that the computing device that created or is otherwise associated with the identified metabase(s) not be involved in processing search operations as further described herein. Thus, the secondary computing device may consult with a manager or index associated with other computing devices to identify metabases with responsive information.

Next at step 640, the secondary processor may search metabases to identify appropriate data set that may potentially have information related to the query. This may involve performing iterative searches that examine results generated by previous searches and subsequently searching additional, previously unidentified metabases to find responsive information that may not have been found during the initial search. Thus the initial metabase search may serve as a starting point for searching tasks that may be expanded based on returned or collected results. Next, at step 650, the returned results may be optionally analyzed for relevance, arranged, and placed in a format suitable for subsequent use (e.g., with another application), or suitable for viewing by a user and reported (step 650).

FIG. 7 presents a system 700 constructed in accordance with the principles of the present invention employing a centralized metabase 760 that may serve multiple computing devices 715-725. For example, as shown, system 700 may include computing devices 715-725, each of which may include a journaling agent (730-740 respectively), a classification agent (745-755 respectively), and centralized metabase 760, and in some embodiments, a manager module 705 with an index 710.

In operation, system 700 may operate substantially similarly to system 300 shown in FIG. 3 with each computing device 715-725 storing classification entries in centralized metabase 760 rather than each computing device having its own dedicated metabase. For example, as shown, data classification agents 745-755 may operate substantially as described herein and communicate results to centralized metabase 760. That is, analyze and process entries within the respective journals associated with journaling agents 730-740, and report results to metabase 760. With this arrangement, the classification agent may provide each metabase entry with an ID tag or other indicia that identifies which computing device 715-725 the entry originated from to facilitate future searches and efficiently designate entry ownership, or other associations between entries and computing devices.

Moreover, each entry to metabase 760 may be assigned a unique identifier for management purposes. As mentioned above, this number may represent the index location or offset of the entry within centralized metabase 760. In some embodiments, entries may be communicated to metabase 760 from the computing devices 715-725 on a rolling basis and may be arranged and formatted for storage by the metabase 760. For example, metabase 760 may receive multiple entries at substantially the same point in time from multiple computing devices 715-725 and may be responsible for queuing and arranging such entries for storage within the metabase 760.

16

In some embodiments, system 700 may include manager module 705 that may be responsible for assigning or removing associations between certain computing devices 715-725 and a particular centralized metabase 760. For example, in accordance with certain system preferences defined in index 710, manager 705 may direct certain computing devices 715-725 to write classification entries to a particular centralized metabase 760. Information indicating an association of the metabase 760 and the computing devices 715-725 may be stored in the index 710. This allows system 700 to reassign resources (globally or locally) to optimize system performance without the need to change device pointers or code associated with each computing device 715-725 that may be affected by a particular reallocation. For example, manager 705 may reassign certain computing devices 715-725 to another metabase by changing a destination address in an appropriate index.

FIG. 8 is flow chart 800 illustrating some of the steps involved in using a centralized metabase with multiple computing devices similar to the one shown in FIG. 7. At step 810, a centralized metabase may be instantiated by a manager module or in accordance with certain system management or provisioning policies. This may involve securing certain processing, storage, and management resources for performing the task, loading certain routines into various memory buffers and informing the management module that the metabase is ready for operation.

Next, at step 820, the management module may review system resources, management policies, operating trends, and other information, for example, to identify computing devices to associate with the instantiated centralized metabase. This may further involve identifying pathways to the metabase from the various computing devices, locating operational policies governing the computing devices and, creating certain logical associations between the centralized metabases and the identified computing devices. These associations, once created, may be stored in an index or database for system management purposes.

After the metabase has been instantiated and associated with computing devices, classification agents within each associated computing device may scan existing files or data on the computing devices or clients (step 825) and populate the centralized metabase as further described herein (step 830). During the scanning process, a computing device identifier or other indicia may be appended or otherwise associated with the entry prior to transmission to the metabase such that each entry in the metabase can be tracked to its associated source computing device (step 840). Next, the centralized metabase may be populated with entries (step 850) and may communicate with the management module to establish and monitor a list of computing devices serviced by the centralized metabase and return to step 830. At this point, the system continues to monitor the associated computing devices for data interactions, which may be reported to the centralized metabase on an ongoing, periodic, or rolling basis.

In certain circumstances, the centralized metabase may need to assimilate or otherwise integrate existing entries with new entries reported by the computing devices. For example, the centralized metabase may become disconnected or unavailable for a period of time and subsequently be required to integrate a large number of queued entries. In this case, the metabase or management module may examine existing metabase entries as described herein and communicate with computing devices to identify: (1) the amount of time the object computer and the metabases have been disconnected, (2) the number of queued entries at the computing devices that need to be processed (for example, entries cached once

US 7,725,671 B2

17

the centralized metabase was inaccessible for write operations), (3) whether there are any duplicative entries, and (4) which entries need to be integrated and in what order of preference (assuming multiple computing devices contain queued entries).

Based on these criteria, the management module or centralized metabase may assimilate the relevant entries into the metabase in the appropriate order until the backlog is eliminated and the system returns to normal operation. If it is determined during this process that certain information is lost to cache overflow, accidental deletion, corruption, or other reasons, the metabase and/or manager module may indicate such a discontinuity with the metadata or index associated with the centralized storage device or management module. In this case, clients, computing devices or other data sources may be rescanned to replace or repair the faulty entries. In other embodiments, the points of discontinuity may be noted and interpolation or other data healing techniques may be employed to provide derived information for the unknown points based on known information.

FIG. 9 presents a system 900 constructed in accordance with the principles of the present invention including a computing device that interacts with a network attached storage device (NAS). As shown, system 900 may include a management module 905 and index 910, computing devices 915-925, each of which may include a journaling agent (945-955 respectively), a classification agent (930-940 respectively), data stores 960 and 965, and metabases 970-980. System 900 may also include NAS device 995 which may include NAS storage device 990 and NAS file system manager 985. Moreover, computing device 925 may be configured to operate as a NAS proxy device supervising the transfer of data to and from NAS device 995.

In operation, system 900 may operate substantially similar to system 300 shown in FIG. 3a with exception of the NAS portion shown on the right-hand side. For example, as shown, data classification agents 930-940 may operate substantially as described herein and communicate results to their respective metabases 970-980. That is, analyze and process entries within the respective journals associated with journaling agents 945-955, and report results to metabases 970-980 which may be supervised in whole or in part by management module 905.

Data from computing device 925 may be journaled and classified using methods similar to those described herein. For example, journaling agent 955 may reside on computing device 925 and track each or certain data interactions between NAS device 995 and external applications. The location of the journaling agent 955 may be external to the NAS device 995 due, at least in part, to its proprietary nature (i.e., a closed system) and the difficulty associated with attempting to run other programs on the NAS device 995 itself.

The NAS portion 995 of system 900 may operate somewhat differently. For example computing device 925 may operate as a NAS proxy for moving data files to and from NAS device 995 using a specialized protocol such as the Network Data Management Protocol (NDMP) that is an open network protocol designed to perform data backups over heterogeneous networks. NDMP may be used to enhance performance by transferring data over a network by separating data and control paths, while maintaining centralized backup administration.

Journaling agent 955 may record any interactions between NAS data and external applications and record those interactions in computing device 925 as described herein. In some embodiments, such a journaling agent may include specialized routines for interpreting and processing data in NAS

18

format. Data classification agent 940 may analyze journal entries and populate metabase 980 initially and periodically as further described herein.

Once initially populated, it may be desired to search the metabases of system 900 for certain information. This is discussed in more detail below in connection with the flow chart 1100 of FIG. 11. In some embodiments, this may be handled by manager 905 or other system process which may initially evaluate any search request and consult index 910 or other information stores to determine which metabases within the system are likely to include responsive information. The results of this evaluation may be provided to the computing device handling the search request and may be in the form of pointers or other indicia or identifiers identifying a metabase such as a metabase ID. This may allow the computing device posing the search request to contact and search the identified metadata directly. In other embodiments manager 905 may process the request and provide substantially complete results to the computing device that submitted the query.

FIG. 10 is flow chart 1000 illustrating some of the steps that may be involved in using the NAS system similar to or the same as the one shown of FIG. 9. At step 1010 a copy operation may be initiated that directs data from computing device to a NAS device. This may involve identifying certain data to moved, for example, based on a data management or storage policy. Other factors that may also be considered may include data size, the last time the data was moved to the NAS device, the file owner, application type, etc.

It will be understood that in some embodiments it may be preferred to use computing device 925 as a NAS proxy that routes data from other network computing devices (not shown) to NAS device 995 with the computing device 925 supervising the data movement using certain specialized transfer programs to assist in the effort (step 1020). As the data is routed through computing device 925, journaling agent 955 may monitor interactions with NAS device 995 and create interaction entries for an interaction journal (step 1030). This may be accomplished by consulting with NAS file manager 985 and identifying which files in NAS 995 that have been involved in a data interaction as further described herein (step 1040). Next, journal entries may be created or updated to reflect data interactions currently detected as previously described herein (step 1050). The interaction journal may then be scanned to analyze the journal records (step 1060) and perform the classification process as further described herein to create metabase entries (step 1070). At this point metabase entries may be assigned an identifier and used to populate metabase 980 (step 1080).

As mentioned above, under certain circumstances, it may be desired to search a system that includes multiple metabases for certain information such as system 900 shown in FIG. 9 whether or not NAS included. FIG. 11 includes a flow chart 1100 illustrating some of the steps that may be performed in searching a multiple metabase system in accordance with certain aspects of the present invention.

Assume, for example, a user wants to locate and copy all data relating to a certain specified criteria such as data relating to a specific marketing project created and edited by a specific group of users over a certain period of time. First, the requestor may formulate such a request through a user interface (not shown) using techniques known in the art and submit the request to the system for processing. This may also be accomplished by an automated computerized process, for example, when the system is performing certain management functions. Next the system may receive and analyze this query (step 1110). In some embodiments, this may be per-

US 7,725,671 B2

19

formed by a computing device configured to support the user interface. In other embodiments, the computing device may simply pass the request to the system where a management module or other system process computing device may perform the analysis. The analysis may include determining characteristics of data in the metabase that may satisfy the selected criteria.

Once the search request or query has been analyzed or parsed, the system may identify all metabases likely to contain records related to relevant data objects based on a query. This may be accomplished by using information obtained from analyzing or parsing the request as well as consulting with a management module that may have a substantially global view of metabases within the system that includes index information or a general overview of the information the metabases contain. After a set of metabases have been identified, the management module or other computing device may perform the search to identify a data set satisfying a query as further described herein and return a set of results (step 1130). At step 1140 the results may optionally be normalized. If normalization is not required, the results may be reported at step 1150. If normalization is desired, the system may analyze the results for both content and completeness. If, based on the returned results, other unsearched metabases are implicated as potentially having information that satisfies the search criteria, those metabases may be searched as well. This process may continue in an iterative fashion until a substantially complete set of results is obtained. Even if no additional metabases are implicated, these results may then be optionally normalized by performing certain functions such as locating and removing duplicative results, identifying network pathways to data objects identified in the search, and formatting or arranging the results for further processing (whether another computing process or for a user). For example, the returned results may be used to locate and retrieve the responsive data objects that may include information located on primary or secondary storage devices within the system or for other purposes as further described herein.

In some embodiments, the systems and methods of the present invention may be employed to identify and track some or all data interactions on a user or group basis. For example, a system administrator or user may wish to record and keep track of all data interactions involving some or all system groups or users. This may include, for example, read and write operations performed on the user's or group's behalf, information and applications used or accessed, viewed web pages, electronic gaming interactions, chat, instant messages, and other communication interactions, multimedia usage, and other Internet or network based electronic interactions as known in the art. Thus, the system identifies, captures, classifies, and may otherwise track user and group interactions with electronic data creating a data store or other repository of these interactions and metadata associated with these interactions. In some embodiments, this repository may serve as a "digital or electronic life record" that effectively chronicles and catalogues some or all user or group interactions with electronic information and data during a given time period as further described herein.

For example, FIG. 11a illustrates a system constructed in accordance with the principles of the present invention that identifies, captures, classifies, and otherwise tracks user and group interactions with electronic data. As shown, the system may generally include computing device 1162, one or more classification agents 1164, one or more journaling agents 1165, metabase 1166, change record 1167, and database 1168.

20

In operation computing device 1162 may be coupled to or interact with various other applications, networks, and electronic information such as, for example multimedia applications 1170, instant messaging/chat applications 1172, network applications 1174 such as an enterprise WAN or LAN, Internet 1176, and gaming applications 1178. It will be understood, however, that these are only examples and that any other network, application, or type of electronic information suitable for the purposes described herein may be added if desired.

Journaling agents 1165 and classification agents 1164 may operate in conjunction with one another to detect and record data interactions as further described herein. For example, each type of electronic data interaction (e.g. - instant messaging, web surfing, Internet search activities, electronic gaming, multimedia usage, etc.) may be identified, captured, classified, and otherwise tracked by a different journaling agent 1165 and classification agent 1164, for example an interaction-specific journaling agent 1165 or classification agent 1164 dedicated to processing a single type of interaction with electronic data. Thus, the system may have a first journaling agent 1165 and a first classification agent 1164 monitoring network traffic on a given network interface (not shown) directed to interactions associated with Internet usage, and a second journaling agent 1165 and a second classification agent 1164 monitoring a different system resource directed to interactions associated with electronic gaming (e.g.—recording and classifying gaming interactions such as recording games played, opponents played, win/loss records, etc.) or directed to interactions associated with use of an Internet browser to "surf" web (e.g.—tracking pages visited, content, use patterns, etc.) In some embodiments, journaling agent 1165 and classification agent 1164 may function as a single module capable of performing some or all functions associated with journaling agent 1165 and a classification agent 1164.

Thus, as a user or group interacts with various types of electronic information, some or all of those interactions may be captured and recorded in database 1168. Change record 1167 and metabase 1166 may record certain aspects of the interactions as further described herein and may represent an interaction by interaction log of the user's computing activities.

For example, in operation, a user of computing device 1162 may interact with certain applications such as multimedia application 1170 and instant messaging application 1172. This may include sending, receiving, viewing and responding to various audio/video files in any suitable format and may include instant, text or email messages. Journaling agent 1165 may detect the interactions between these applications and computing device 1162 and classification agent 1164 may classify and record information (e.g., metadata) associated with these interactions in metabase 1166 as further described herein.

Moreover, in some embodiments, some or all the content being exchanged or otherwise associated with these interactions may be captured and stored in database 1168 or other storage locations in the system. This may include capturing screen shots or summaries of information exchanges during data interactions. For example, the system may download all content associated with web pages viewed thus being able to recreate the original page content and interaction without access to the original or source version of the page on the Internet or other network. This may be advantageous, for example, if a user wishes to interact with content associated with a previous interaction when that content is no longer available, as is common with web pages and other network

US 7,725,671 B2

21

resources over time. As another example, the system may also capture or otherwise store data associated with other interactions, for example chat transcripts, video game replays, search queries, search results, and associated search content, songs accessed, movies accessed, stored songs and movies, in addition to metadata, etc.

Moreover, in some embodiments, specialized classification agents may be employed for some or all of the applications that a user or administrator desires to track and record. For example, the multimedia and instant messaging applications described above may each have a dedicated classification agent that analyzes journal records to create entries for metabase 1166. Further still, each classification agent may have its own associated metabase and/or repository for source data (not shown), so application histories and content may be quickly indexed and searched. In other embodiments, however, a “universal” classification agent may be used that recognizes the application type (e.g., based on the journaling agent entries) and process interactions accordingly (which may include routing metadata to one or more specialized metabases).

As shown in FIG. 11a, computing device 1162 may also interact with various network applications 1174 such as LAN or WAN applications. These may include interaction with certain distributed programs such as Microsoft Word or Outlook. Users may also interact with Internet 1176 and download various web pages and other information. In accordance with an aspect of the present invention, interactions with these networks/applications may also be journaled as described above with certain information regarding these interactions stored in metabase 1166. Portions of exchanged content may also be stored in database 1166. For example, Word documents, emails, web pages, web addresses and HTML content may be captured and stored on database 1168 such that it substantially represents a record of all user interactions with computing device 1162, or other system devices. For example, user interactions may be recorded with respect to any identified user based on identifiers and tracked at any network computing device.

Thus, if desired a user may retrieve captured data and review or replay certain data exchanges or save such records for future reference. For example, a user may store all instant messaging interactions for replay or transmission to another. In some instances, it may be desirable to not record certain interactions, such as personal or private information. In some embodiments, this may be accomplished by “disabling” the appropriate classification agent for a certain period of time, etc.

Likewise, interactions with gaming applications (network or stand alone) may also be recorded stored with appropriate information stored in database 1168 and metabase 1166. Thus, a user may have the ability to retrieve, replay and transmit certain saved gaming sequences to third parties.

In some embodiments, database 1168 may become large and thus some information stored thereon may be moved to single instance storage from database 1168 with a pointer placed in the logical address of the instanced information (not shown). This may be performed as a memory saving measure as at least some of the entries in database 1168 are likely to be duplicative.

Some of the steps associated with the method generally described above are illustrated in chart 1200 of FIG. 12 and may include the following. At the outset, a group or user of interest may be identified based on certain user related information or other network characteristics (step 1210). Such characteristics may include Active Directory privileges, network login, machine ID, or certain biometrics associated with

22

a user or group member. These characteristics may be combined together or associated with one another to create a user or group profile. Such profiles may be stored in a database or index within a management module of the system and may be used as classification definitions within the system. When it is desired to identify or classify data items associated with a particular interaction, the system may compare certain attributes of the data involved in a detected interaction and associate that interaction with a particular group or user based on profile information (step 1220).

Such associations may be stored in a metabase created to keep track of user or group interactions. Thus, in one embodiment, the metabase essentially represents a list of all data interaction for a particular group or user. If desired, a list or copy of all the data items touched (e.g., interacted with) by a group or user may be quickly obtained.

In operation, the system may, through the use of a journaling agent or the like, monitor data interactions for a particular computing device as described herein. The interactions may be analyzed by a classification agent as described herein and associated with one or more profiles (step 1230). The association may be recorded in an identified metabase(s) that keeps track of a user’s or group’s interactions (step 1240) which may include references to the data object(s) identified, the attributes compared, and the basis for the association, etc. As discussed herein, the journaling agent may continue to monitor data interactions throughout operation, so that each metabase is updated and continues to accurately represent the data touched by a particular group or user. The identified metabases are associated with an identified group or user (step 1250), such as by storing an indication of the association in an index.

FIG. 13 presents a system 1300 constructed in accordance with the principles of the present invention for communicating metadata and/or data objects between two or more computing devices. As shown, system 1300 may generally include first and second computing devices 1310 and 1320, respectively, associated data stores 1330 and 1340, and metabases 1350 and 1360. Computing devices in system 1300 may store data objects and metadata in their respective metabases and data stores as further described herein. In certain situations, however, it may be desired to transfer certain metadata between metabases 1350 and 1360 and certain data objects between data stores 1330 and 1340. This may be desirable for example, to move certain data from one computing device to another, to recreate a certain application at another location, or to copy or backup certain data objects and associated metadata.

FIG. 14 presents a flow chart 1400 illustrating some of the steps associated with moving data between the computing devices described above. First, at step 1410, data objects and/or associated metadata may be identified for movement from one computing device to another. This may be accomplished by forming a query for certain data, such as a search for data that may be to be moved or copied pursuant to a data management or storage policy, or in response to a request to move data relating to certain processes or applications from one computing device to another, for any other suitable purpose such as disaster recovery, resource reallocation or reorganization, load balancing, etc.

At step 1420, the query may be analyzed and a first data store associated with a first computer may be searched for data objects satisfying the search criteria. Data objects identified during this process may then be transferred to a second data store associated with a second computing device (step 1430). Metadata associated with the transferred data objects may also be identified in a first metabase associated with the

US 7,725,671 B2

23

first computing device and transferred to an appropriate second metabase associated with the second computing device (step 1440). Such a transfer may involve copying data objects and metadata from one data store and metabase to another, or in some embodiments, may involve migrating the data from its original location to a second location and leaving a pointer or other reference to the second location so the moved information may be quickly located from information present at the original location.

FIG. 15 illustrates a one arrangement of resources in a computing network in accordance with the principles of the present invention. As shown, storage operation cell 1550 may generally include a storage manager 1501, a data agent 1595, a media agent 1505, a storage device 1515, and, in some embodiments, may include certain other components such as a client 1585, a data or information store 1590, database 1511, jobs agent 1520, an interface module 1525, and a management agent 1530. Such system and elements thereof are exemplary of a modular storage system such as the CommVault QiNetix system, and also the CommVault GALAXY backup system, available from CommVault Systems, Inc. of Oceanport, N.J., and further described in U.S. patent application Ser. No. 09/610,738 which is incorporated herein by reference in its entirety.

A storage operation cell, such as cell 1550, may generally include combinations of hardware and software components associated with performing storage operations on electronic data. According to some embodiments of the invention, storage operations cell 50 may be related to backup cells and provide some or all of the functionality of backup cells as described in application Ser. No. 09/354,058, which is hereby incorporated by reference in its entirety. However, in certain embodiments, storage operation cells may also perform additional types of storage operations and other types of storage management functions that are not generally offered by backup cells.

In accordance with certain embodiments of the present invention, additional storage operations performed by storage operation cells may include creating, storing, retrieving, and migrating primary storage data (e.g., 1590) and secondary storage data (which may include, for example, snapshot copies, backup copies, HSM copies, archive copies, and other types of copies of electronic data) stored on storage devices 1515. In some embodiments, storage operation cells may also provide one or more integrated management consoles for users or system processes to interface with in order to perform certain storage operations on electronic data as further described herein. Such integrated management consoles may be displayed at a central control facility or several similar consoles distributed throughout multiple network locations to provide global or geographically specific network data storage information.

In some embodiments, storage operations may be performed according to various storage preferences, for example as expressed by a user preference or storage policy. A storage policy is generally a data structure or other information source that includes a set of preferences and other storage criteria associated with performing a storage operation. The preferences and storage criteria may include, but are not limited to, a storage location, relationships between system components, network pathway to utilize, retention policies, data characteristics, compression or encryption requirements, preferred system components to utilize in a storage operation, and other criteria relating to a storage operation. Thus, a storage policy may indicate that certain data is to be stored in a specific storage device, retained for a specified period of time before being aged to another tier of secondary

24

storage, copied to secondary storage using a specified number of streams, etc. A storage policy may be stored in a storage manager database 1511, to archive media as metadata for use in restore operations or other storage operations, or to other locations or components of the system.

A schedule policy may specify when to perform storage operations and how often and may also specify performing certain storage operations on sub-clients of data and how to treat those sub-clients. A sub-client may represent static or dynamic associations of portions of data of a volume and are typically mutually exclusive. Thus, a portion of data may be given a label and the association is stored as a static entity in an index, database or other storage location used by the system. Sub-clients may also be used as an effective administrative scheme of organizing data according to data type, department within the enterprise, storage preferences, etc.

For example, an administrator may find it preferable to separate e-mail data from financial data using two different sub-clients having different storage preferences, retention criteria, etc. Storage operation cells may contain not only physical devices, but also may represent logical concepts, organizations, and hierarchies. For example, a first storage operation cell 1550 may be configured to perform a first type of storage operations such as HSM operations, which may include backup or other types of data migration, and may include a variety of physical components including a storage manager 1501 (or management agent 1530), a media agent 1505, a client component 1585, and other components as described herein. A second storage operation cell may contain the same or similar physical components, however, it may be configured to perform a second type of storage operations such as SRM operations, and may include as monitoring a primary data copy or performing other known SRM operations.

Thus, as can be seen from the above, although the first and second storage operation cells are logically distinct entities configured to perform different management functions (i.e., HSM and SRM respectively), each cell may contain the same or similar physical devices in both storage operation cells. Alternatively, in other embodiments, different storage operation cells may contain some of the same physical devices and not others. For example, a storage operation cell 1550 configured to perform SRM tasks may contain a media agent 1505, client 1585, or other network device connected to a primary storage volume, while a storage operation cell 1550 configured to perform HSM tasks may instead include a media agent 1505, client 1585, or other network device connected to a secondary storage volume and not contain the elements or components associated with and including the primary storage volume. These two cells, however, may each include a different storage manager 1501 that coordinates storage operations via the same media agents 1505 and storage devices 1515. This "overlapping" configuration allows storage resources to be accessed by more than one storage manager 1501 such that multiple paths exist to each storage device 1515 facilitating failover, load balancing and promoting robust data access via alternative routes.

Alternatively, in some embodiments, the same storage manager 1501 may control two or more cells 1550 (whether or not each storage cell 1550 has its own dedicated storage manager 100). Moreover, in certain embodiments, the extent or type of overlap may be user-defined (through a control console (not shown)) or may be automatically configured to optimize data storage and/or retrieval.

Data agent 1595 may be a software module or part of a software module that is generally responsible for copying, archiving, migrating, and recovering data from client com-

US 7,725,671 B2

25

puter **1585** stored in an information store **1590** or other memory location. Each client computer **1585** may have at least one data agent **1595** and the system can support multiple client computers **1585**. In some embodiments, data agents **1595** may be distributed between client **1585** and data storage manager **1501** (and any other intermediate components (not shown)) or may be deployed from a remote location or its functions approximated by a remote process that performs some or all of the functions of data agent **1595**.

Embodiments of the present invention may employ multiple data agents **1595** each of which may backup, migrate, and recover data associated with a different application. For example, different individual data agents **1595** may be designed to handle Microsoft Exchange data, Lotus Notes data, Microsoft Windows 2000 file system data, Microsoft Active Directory Objects data, and other types of data known in the art. Other embodiments may employ one or more generic data agents **1595** that can handle and process multiple data types rather than using the specialized data agents described above.

If a client computer **1585** has two or more types of data, one data agent **1595** may be required for each data type to copy, archive, migrate, and restore the client computer **1585** data. For example, to backup, migrate, and restore all of the data on a Microsoft Exchange 2000 server, the client computer **1585** may use one Microsoft Exchange 2000 Mailbox data agent **1595** to backup the Exchange 2000 mailboxes, one Microsoft Exchange 2000 Database data agent **1595** to backup the Exchange 2000 databases, one Microsoft Exchange 2000 Public Folder data agent **1595** to backup the Exchange 2000 Public Folders, and one Microsoft Windows 2000 File System data agent **1595** to backup the client computer's **1585** file system. These data agents **1595** would be treated as four separate data agents **1595** by the system even though they reside on the same client computer **1585**.

Alternatively, other embodiments may use one or more generic data agents **1595**, each of which may be capable of handling two or more data types. For example, one generic data agent **1595** may be used to back up, migrate and restore Microsoft Exchange 2000 Mailbox data and Microsoft Exchange 2000 Database data while another generic data agent may handle Microsoft Exchange 2000 Public Folder data and Microsoft Windows 2000 File System data, etc.

Data agents **1595** may be responsible for arranging or packing data to be copied or migrated into a certain format such as an archive file which is discussed in more detail in connection with FIG. 16 herein. Nonetheless, it will be understood this represents only one example and any suitable packing or containerization technique or transfer methodology may be used if desired. Such an archive file may include a list of files or data objects copied in metadata, the file and data objects themselves. Moreover, any data moved by the data agents may be tracked within the system by updating indexes associated appropriate storage managers or media agents.

Generally speaking, storage manager **1501** may be a software module or other application that coordinates and controls storage operations performed by storage operation cell **1550**. Storage manager **1501** may communicate with some or all elements of storage operation cell **1550** including client computers **1585**, data agents **1595**, media agents **1505**, and storage devices **1515**, to initiate and manage system backups, migrations, and data recovery.

Storage manager **1501** may include a jobs agent **1520** that monitors the status of some or all storage operations previously performed, currently being performed, or scheduled to be performed by storage operation cell **1550**. Jobs agent **1520** may be communicatively coupled with an interface agent

26

1525 (typically a software module or application). Interface agent **1525** may include information processing and display software, such as a graphical user interface ("GUI"), an application program interface ("API"), or other interactive interface through which users and system processes can retrieve information about the status of storage operations. Through interface **1525**, users may optionally issue instructions to various storage operation cells **1550** regarding performance of the storage operations as described and contemplated by the present invention. For example, a user may modify a schedule concerning the number of pending snapshot copies or other types of copies scheduled as needed to suit particular needs or requirements. As another example, a user may employ the GUI to view the status of pending storage operations in some or all of the storage operation cells in a given network or to monitor the status of certain components in a particular storage operation cell (e.g., the amount of storage capacity left in a particular storage device).

Storage manager **1501** may also include a management agent **1530** that is typically implemented as a software module or application program. In general, management agent **1530** provides an interface that allows various management components **1501** in other storage operation cells **1550** to communicate with one another. For example, assume a certain network configuration includes multiple cells **1550** adjacent to one another or otherwise logically related in a WAN or LAN configuration (not shown). With this arrangement, each cell **1550** may be connected to the other through each respective interface agent **1525**. This allows each cell **1550** to send and receive certain pertinent information from other cells **1550** including status information, routing information, information regarding capacity and utilization, etc. These communication paths may also be used to convey information and instructions regarding storage operations.

For example, a management agent **1530** in first storage operation cell **1550** may communicate with a management agent **1530** in a second storage operation cell **1550** regarding the status of storage operations in the second storage operation cell. Another illustrative example includes the case where a management agent **1530** in first storage operation cell **1550** communicates with a management agent **1530** in a second storage operation cell to control the storage manager **1501** (and other components) of the second storage operation cell via the management agent **1530** contained in the storage manager **100**.

Another illustrative example is the case where management agent **130** in the first storage operation cell **1550** communicates directly with and controls the components in the second storage management cell **1550** and bypasses the storage manager **1501** in the second storage management cell. If desired, storage operation cells **1550** can also be organized hierarchically such that hierarchically superior cells control or pass information to hierarchically subordinate cells or vice versa.

Storage manager **1501** may also maintain an index, a database, or other data structure **1511**. The data stored in database **1511** may be used to indicate logical associations between components of the system, user preferences, management tasks, media containerization and data storage information or other useful data. For example, the storage manager **1501** may use data from database **1511** to track logical associations between media agent **1505** and storage devices **1515** (or movement of data as containerized from primary to secondary storage).

Generally speaking, a media agent, which may also be referred to as a secondary storage computing device, **1505** may be implemented as software module that conveys data, as

US 7,725,671 B2

27

directed by storage manager **1501**, between a client computer **1585** and one or more storage devices **1515** such as a tape library, a magnetic media storage device, an optical media storage device, or any other suitable storage device. In one embodiment, secondary computing device **1505** may be communicatively coupled with and control a storage device **1515**. A secondary computing device **1505** may be considered to be associated with a particular storage device **1515** if that secondary computing device **1505** is capable of routing and storing data to particular storage device **1515**.

In operation, a secondary computing device **1505** associated with a particular storage device **1515** may instruct the storage device to use a robotic arm or other retrieval means to load or eject a certain storage media, and to subsequently archive, migrate, or restore data to or from that media. Secondary computing device **1505** may communicate with a storage device **1515** via a suitable communications path such as a SCSI or fiber channel communications link. In some embodiments, the storage device **1515** may be communicatively coupled to a data agent **105** via a Storage Area Network ("SAN").

Each secondary storage computing device **1505** may maintain an index, a database, or other data structure **1506** which may store index data generated during backup, migration, and restore and other storage operations as described herein. For example, performing storage operations on Microsoft Exchange data may generate index data. Such index data provides a secondary computing device **1505** or other external device with a fast and efficient mechanism for locating data stored or backed up. Thus, in some embodiments, a secondary storage computing device index **1506**, or a storage manager database **1511**, may store data associating a client **1585** with a particular secondary computing device **1505** or storage device **1515**, for example, as specified in a storage policy, while a database or other data structure in secondary computing device **1505** may indicate where specifically the client **1585** data is stored in storage device **1515**, what specific files were stored, and other information associated with storage of client **1585** data. In some embodiments, such index data may be stored along with the data backed up in a storage device **1515**, with an additional copy of the index data written to index cache in a secondary storage device. Thus the data is readily available for use in storage operations and other activities without having to be first retrieved from the storage device **1515**.

Generally speaking, information stored in cache is typically recent information that reflects certain particulars about operations that have recently occurred. After a certain period of time, this information is sent to secondary storage and tracked. This information may need to be retrieved and uploaded back into a cache or other memory in a secondary computing device before data can be retrieved from storage device **1515**. In some embodiments, the cached information may include information regarding format or containerization of archive or other files stored on storage device **1515**.

In some embodiments, certain components may reside and execute on the same computer. For example, in some embodiments, a client computer **1585** such as a data agent **1595**, or a storage manager **1501** coordinates and directs local archiving, migration, and retrieval application functions as further described in U.S. patent application Ser. No. 09/610, 738. This client computer **1585** can function independently or together with other similar client computers **1585**.

Moreover, as shown in FIG. **15**, clients **1585** and secondary computing devices **1505** may each have associated metabases (**1525** and **1560**, respectively). However in some embodiments each "tier" of storage, such as primary storage, sec-

28

ondary storage, tertiary storage, etc., may have multiple metabases or a centralized metabase, as described herein. For example, in FIG. **15**, rather than a separate metabase **1525** associated with each client **1585**, the metabases on this storage tier may be centralized as discussed further herein. Similarly, second and other tiers of storage may have either centralized or distributed metabases. Moreover, mixed architectures systems may be used if desired, that may include a first tier centralized metabase system coupled to with a second tier storage system having distributed metabases and vice versa, etc.

Moreover, in operation, a storage manager **1501** or other management module may keep track of certain information that allows the storage manager to select, designated or otherwise identify metabases to be searched in response to certain queries as further described herein. Movement of data between primary and secondary storage may also involve movement of associated metadata and other tracking information as further described herein.

FIG. **16** is a diagram illustrating one arrangement of data that may be used in constructing an archive file according to one aspect of the invention. As shown, archive file **1600** may include header section **1610**, index section **1620** and payload section **1630**. Such an archive file may be constructed by a data agent at a client computing device when migrating data, for example, from primary to secondary storage, primary storage to other primary storage, etc. The payload section **1610** may include the data objects that are to be moved from a first location to a second location within the system (e.g., primary to secondary storage). These data objects may be identified by a data agent and designated to be moved pursuant to a storage preference such as a storage policy, a user preference, etc. Header **1610** may include routing and path information that identifies the origin and destination of the payload data and may include other information such as a list of files copied, checksums, etc. Index section **1620** may include certain other information regarding the payload data objects such as size, file type, and any offset or other logical indexing information that may be tracked by a storage management component or other component previously managing the data objects in the payload.

In some embodiments, storage managers may index information regarding archive files and related payload by time and storage on certain media so the archive files can be quickly located and/or retrieved. For example, it may be desired to identify certain data based on a query. The query may be analyzed and a certain time frame of interest may be identified. The system may use this information as a basis for a query search of certain index information (e.g., only search for records concerning operations that occurred during a specific time). This streamlines the search and retrieval process by narrowing the universe of data needs to be searched to locate responsive information.

FIG. **17** presents a flow chart **1700** that illustrates some of the steps that may be performed in moving data from primary storage to other storage devices within the system. First, at step **1702**, a query seeking certain data may be identified. The query may include aspects of data such as a schedule policy, storage policy, storage preference or other preference. The query may be analyzed and a primary metabase searched to identify data objects that satisfy the query (step **1704**). This may include parsing the query into constituent parts and analyzing each part alone or in combination with other portions as part of the evaluation process. At step **1706**, it may be determined, whether data objects satisfying the query are to be copied to other primary storage devices, to secondary storage devices or both (pursuant to a storage policy, etc.).

US 7,725,671 B2

29

If at least some data objects satisfying the search criteria are to be copied to other primary storage devices, those data objects may be identified as further described herein and the target primary storage device(s) identified. This may involve consulting a storage policy or storage manager to determine the destination point. In some embodiments, destination may be determined dynamically, such that it is selected based on certain system preferences or optimization routines that select a storage device based on storage capacity, availability, data paths to the destination, etc.

At step 1708 the identified data objects may be copied from primary storage of a first computing device (the source) to primary storage of a second computing device (the target or destination). Any metadata associated with the first computing device describing the copied data may also be copied to a metabase associated with the second computing device such that this description information is not abandoned or lost, but rather travels with the copied data for subsequent use (step 1710).

Next, at step 1712, it may be determined whether the copied data objects and associated metadata are to be deleted from the source computing device. For example, this may be done in order to free storage space on the source computer or in accordance with certain data aging or migration criteria. If it is decided to delete the data objects (and associated metadata) the memory locations which include the data may be erased or designated for overwrite (step 1714 and 1716).

In some embodiments the data objects may be deleted but certain metadata may be retained. If it is decided not delete the data objects, the data is retained and an index in an associated storage manager may be updated (step 1718), for example by updating an index to reflect a new location, data object status, any changes, etc., and return to step 1702. In other embodiments, if data is deleted from the system, for example, a user permanently deletes certain data from an application, that associated data may also be deleted from both primary and secondary storage devices and associated metabases to free storage space within the system.

Returning to step 1706, it is also determined whether certain data objects currently stored in primary storage are to be migrated to one or more secondary storage devices. If so, an archive file similar to the one described in FIG. 16 or other data structure suitable for transport may be constructed or created by the source computing device with identified data objects placed in the payload section and header and index information added (step 1722). Data may be moved from primary to secondary storage in predefined chunks which are constructed from such archive files, for example, using a data pipe, such as the data pipe described in Pat. No. 6,418,478 titled PIPELINED HIGH SPEED DATA TRANSFER MECHANISM.

Next, at step 1724 one or more target secondary storage devices may be identified. This may involve consulting a storage policy or storage manager to determine the destination point. In some embodiments, destination may be determined dynamically, such that it is selected based on certain system preferences or optimization routines that select a storage device based on storage capacity, availability, data paths to the destination, etc. Once the secondary storage device(s) are identified, the archive files may be routed to a media agent, storage manager, or other system component that supervises the transfer to the target secondary storage device (steps 1724 and 1728). This may involve selecting and appropriate data transfer route and ensuring the proper resources and are available (e.g., bandwidth) such that the data may be copied with a certain period of time. Supervision may further include parsing a copy operation into several portions with each por-

30

tion being transferred by certain media agent or other resources, etc., to meet system or transfer requirements (e.g., a time window).

Next, the appropriate media within the target storage device may be identified (step 1730) and the archive files may be transferred from the media management device to the secondary storage device (step 1732). Such media may be selected from available media already associated with a similar data transfer or may be selected and reserved from an available media pool or scratch pool within the storage device. During or after the transfer, a media agent index or storage manager index associated with the secondary storage device may be updated to reflect the transfer (step 1733). This may include copying the appropriate management files to the media management index such as offset, media ID file name or other management information.

At step 1734, any metadata stored in a first metabase associated with the transferred data objects may also be transferred and used to update a second metabase associated with the target secondary storage device. Such metadata may be copied from the first metabase to the second metabase using network transmission resources. In some embodiments, the metadata in the first metabase may be deleted after it is confirmed the metadata has been copied to the second metabase. In other embodiments, the metadata may remain in both first and second metabases.

At step 1736, it may be determined whether the data objects transferred from the primary storage device are to be deleted. If so, the data objects and associated metadata in a first metabase may be erased or otherwise designated for overwrite (steps 1738 and 1740). In some cases, a pointer or other reference such as a file stub may be left in the original data location.

FIG. 18 presents a generalized block diagram of a hierarchically organized group of storage operation cells in a system to perform storage operations on electronic data in a computer network in accordance with an embodiment of the present invention. It will be understood that although the storage operation cells generally depicted in FIG. 18 have different reference numbers than the storage operation cell 1550 shown in FIG. 15, these cells may be configured the same as or similar to the storage cell 1550 as depicted in FIG. 15.

As shown, the system illustrated in FIG. 18 may include a master storage manager component 1835 and various other storage operations cells. As shown, the illustrative embodiment in FIG. 18 includes a first storage operation cell 1840, a second storage operation cell 1845, a third storage operation cell 1850, a fourth storage operation cell 1855, and may be extended to include nth storage operation cell, if desired (not shown). However, it will be understood this illustration is only exemplary and that fewer or more storage operation cells may be present or interconnected differently if desired.

Storage operation cells, such as the ones shown in FIG. 18 may be communicatively coupled and hierarchically organized. For example, a master storage manager 1835 may be associated with, communicate with, and direct storage operations for a first storage operation cell 1840, a second storage operation cell 1845, a third storage operation cell 1850, and fourth storage operation cell 1855. In some embodiments, the master storage manager 1835 may not be part of any particular storage operation cell. In other embodiments (not shown), master storage manager 1835 may itself be part of a certain storage operation cell. This logical organization provides a framework in which data objects, metadata and other management data may be hierarchically organized and associated with appropriate devices components (e.g., storage devices).

US 7,725,671 B2

31

The storage operation cells may be configured in any suitable fashion, including those which involve distributed or centralized metabases. For example, storage operation cell **1840** may include a centralized primary storage metabase and a centralized secondary storage metabase, storage operation cell **1845** may include a centralized primary storage metabase and multiple secondary storage metabases, storage operation cell **1850** may include multiple primary storage metabases and a centralized secondary storage metabase, and storage operation cell **1855** may include multiple primary storage metabases and multiple secondary storage metabases (not shown). However, it will be understood that this is merely illustrative, and any other suitable configuration may be used if desired.

Thus, in operation, master storage manager **1835** may communicate with a management agent of the storage manager of the first storage operation cell **1840** (or directly with the other components of first cell **1840**) with respect to storage operations performed in the first storage operation cell **1840**. For example, in some embodiments, master storage manager **1835** may instruct the first storage operation cell **1840** with certain commands regarding a desired storage operation such as how and when to perform particular storage operations including the type of operation and the data on which to perform the operation.

Moreover, metabases associated with each storage operation cell may contain information relating to data and storage operations as described herein. In some embodiments, master storage manager **1835** may include a master metabase index or database (not shown) that reflects some or all of the meta-data information from the hierarchically subordinate storage operation cells within the system. This allows the system to consult the master storage index or database for information relating to data within those storage operation cells rather than requiring each cell be contacted or polled directly for such information.

In other embodiments, master storage manager **1835** may track the status of its associated storage operation cells, such as the status of jobs, system components, system resources, and other items, by communicating with manager agents (or other components) in the respective storage operation cells. Moreover, master storage manager **1835** may track the status of its associated storage operation cells by receiving periodic status updates from the manager agents (or other components) in the respective cells regarding jobs, system components, system resources, and other items. For example, master storage manager **1835** may use methods to monitor network resources such as mapping network pathways and topologies to, among other things, physically monitor storage operations and suggest, for example, alternate routes for storing data as further described herein.

In some embodiments, master storage manager **1835** may store status information and other information regarding its associated storage operation cells and other system information in an index cache, database or other data structure accessible to manager **1835**. A presentation interface included in certain embodiments of master storage manager **1835** may access this information and present it to users and system processes with information regarding the status of storage operations, storage operation cells, system components, and other information of the system.

In some embodiments, master storage manager **1835** may store and/or track metadata and other information regarding its associated storage operation cells and other system information in an index cache, database or other data structure accessible to manager **1835**. Thus, during a search procedure as further described herein, queries can be directed to a spe-

32

cific storage operation cell or cells based on the cell's function, past involvement, routing or other information maintained within the storage manager or other management component.

As mentioned above, storage operation cells may be organized hierarchically. With this configuration, storage operation cells may inherit properties from hierarchically superior storage operation cells or be controlled by other storage operation cells in the hierarchy (automatically or otherwise). Thus, in the embodiment shown in FIG. **18**, storage operation cell **1845** may control or is otherwise hierarchically superior to storage operation cells **1850** and **1855**. Similarly, storage operation cell **1850** may control storage operation cells **1855**. Alternatively, in some embodiments, storage operation cells may inherit or otherwise be associated with storage policies, storage preferences, storage metrics, or other properties or characteristics according to their relative position in a hierarchy of storage operation cells.

Storage operation cells may also be organized hierarchically according to function, geography, architectural considerations, or other factors useful or desirable in performing storage operations. For example, in one embodiment, storage operation cell **1840** may be directed to create snapshot copies of primary copy data, storage operation cell **1845** may be directed to create backup copies of primary copy data or other data. Storage operation cell **1840** may represent a geographic segment of an enterprise, such as a Chicago office, and storage operation cell **1845** may represent a different geographic segment, such as a New York office. In this example, the second storage operation cells **1845**, **1850** and **1855** may represent departments within the New York office. Alternatively, these storage operation cells could be further divided by function performing various types of copies for the New York office or load balancing storage operations for the New York office.

As another example, and as previously described herein, different storage operation cells directed to different functions may also contain the same or a subset of the same set of physical devices. Thus, one storage operation cell in accordance with the principles of the present invention may be configured to perform SRM operations and may contain the same, similar or a subset of the same physical devices as a cell configured to perform HSM or other types of storage operations. Each storage operation cell may, however, share the same parent or, alternatively, may be located on different branches of a storage operation cell hierarchy tree. For example, storage operation cell **1845** may be directed to SRM operations whereas storage operation cell **1855** may be directed to HSM operations. Those skilled in the art will recognize that a wide variety of such combinations and arrangements of storage operation cells are possible to address a broad range of different aspects of performing storage operations in a hierarchy of storage operation cells.

In some embodiments, hierarchical organization of storage operation cells facilitates, among other things, system security and other considerations. For example, in some embodiments, only authorized users may be allowed to access or control certain storage operation cells. For example, a network administrator for an enterprise may have access to many or all storage operation cells including master storage manager **1835**. But a network administrator for only the New York office, according to a previous example, may only have access to storage operation cells **1845-1855**, which form the New York office storage management system.

Moreover, queries performed by the system may be subject to similar restrictions. For example, depending on access privileges, users may be limited or otherwise excluded from

US 7,725,671 B2

33

searching a certain cell or cells. For example, a user may be limited to searching information in cells or metabases within the system that are unrestricted or to those which specific access rights have been granted. For example, certain users may not have privileges to all information within the system. Accordingly, in some embodiments, as a default setting, users may have access privileges to information in cells that they interact with. Thus, confidential and sensitive information may be selectively restricted except only to certain users with express privileges (e.g., financial or legal information, etc.). For example, certain classification information within the metabases in the system may be restricted and therefore accessed only by those with the proper privileges.

Other restrictions on search criteria may include the scope of the search. For example, in a large network with many storage cells may require dedicating significant amounts of resources to perform go global or comprehensive searches. Thus, if a certain resource threshold is exceeded by a proposed search, the system may prompt that search to be modified or otherwise cancelled.

In other embodiments master storage manager 1835 may alert a user such as a system administrator when a particular resource is unavailable or congested. For example, a particular storage device might be full or require additional media. For example, a master storage manager may use information from an HSM storage operation cell and an SRM storage operation cell to present indicia or otherwise alert a user or otherwise identify aspects of storage associated with the storage management system and hierarchy of storage operation cells.

Alternatively, a storage manager in a particular storage operation cell may be unavailable due to hardware failure, software problems, or other reasons. In some embodiments, master storage manager 1835 (or another storage manager within the hierarchy of storage operation cells) may utilize the global data regarding its associated storage operation cells to suggest solutions to such problems when they occur (or act as a warning prior to occurrence). For example, master storage manager 1835 may alert the user that a storage device in a particular storage operation cell is full or otherwise congested, and then suggest, based on job and data storage information contained in its database, or associated metabase, or an alternate storage device. Other types of corrective actions based on such information may include suggesting an alternate data path to a particular storage device, or dividing data to be stored among various available storage devices as a load balancing measure or to otherwise optimize storage or retrieval time. In some embodiments, such suggestions or corrective actions may be performed automatically, if desired. This may include automatically monitoring the relative health or status of various storage operation cells and searching for information within the cells of the system relating to systems or resource performance within that cell (e.g., index, metabase, database, etc.) for use in diagnostics or for suggesting corrective action.

In alternate embodiments, HSM and SRM components may be aware of each other due to a common database or metabase of information that may include normalized data from a plurality of cells. Therefore, in those embodiments there is no need for such information to pass through a master storage manager as these components may be able to communicate directly with one another. For example, storage operation cell 1845 may communicate directly with storage operation cell 1855 and vice versa. This may be accomplished through a direct communications link between the two or by passing data through intermediate cells.

34

Moreover, in some embodiments searches may be performed across a numerous storage cells within the hierarchy. For example, a query may be posed to master storage manager 1835 that may pass the query down through the hierarchy from cells 1840 to 1845 to 1850 and 1855, etc. This may be accomplished by passing the query from one manager component of each cell to another, or from one data classification agent to another, one metabase to another etc. The results may be passed upward through the hierarchy and compiled with other results such that master storage manager 1835 has a complete set of results to report. In other embodiments, each storage manager cell may report results directly to the requestor or to a designated location.

FIG. 19 presents a flow chart 1900 that illustrates some of the steps that may be involved in performing searches for data objects across systems that include multiple primary and secondary storage devices. First, at step 1905, a query seeking certain data may be identified (e.g., from a storage policy, user preference, other process, etc.). The query may be analyzed to identify system components, such as clients potentially having information such as certain data objects or metadata that may satisfy the query (e.g., by excluding certain clients that are unlikely to have data being sought based on certain query parameters such as location, time frame, client or other component, department, application type, or any other criteria used to classify data as described herein, etc. (step 1910)). Results may be presented based on a confidence factor indicating the likelihood that the results meet the specified parameters. For example, results substantially satisfying most or all criteria may be listed first with the confidence factors provided based on a percentage of the criteria satisfied (e.g., a query that returned results having three out of four criteria satisfied may be represented with a 75% confidence factor etc.). Less relevant results may be listed subsequently with the confidence factor provided based on any suitable relevant factor such as number of parameters satisfied, how close the match is, etc.

The search process may further involve consulting one or more indexes associated with the clients to identify where responsive data objects or other copies of client data, etc., may be located within the system. At step 1915, it may be determined whether client data objects satisfying the query are located in primary storage, secondary storage, or both (e.g., based on index information in a storage manager). This may be based on polling various storage managers or a master storage manager that includes information the covers or represents whole system or the portion of system specified for search.

If it is determined that responsive data objects are only located on client(s) in primary storage, that client may be added to the list of clients to be searched (step 1955). If it is determined that responsive data objects are located in secondary storage devices (or other primary storage locations that may be identified), the system may consult a storage manager index to identify archive files (or other files) based on certain query parameters such as a specified point in time, origination point, etc., or on index data stored in a storage manager index identifying archive files or other file associated with the data objects.

Next at step 1920, storage managers may be consulted to identify responsive archive files. At step 1925, media management components that may have handled responsive data objects are identified. This may be based on information retrieved from the storage manager index regarding archive files, e.g., an association of archive files with media agents and media items. It may then be determined whether the identified media management components have metadata

US 7,725,671 B2

35

relating to the identified archive files available readily available in an index cache (step 1930).

This may be accomplished by searching for reference information relating to the identified archive files. If such information is already present in the cache, responsive data objects may be identified and retrieved using the index cache information, which may include, offsets and any file identifiers, etc., by the media management component, and the system may proceed to step 1940 (determine whether another media management component needs to be analyzed).

If not, the index information may need to be loaded from the secondary storage device so archive files may be retrieved and accessed. This may involve identifying the particular media on which the index data is stored and upload it to the media management component cache (step 1935). In some embodiments, a master storage manager or other component with information relating to files may be consulted to identify media containing the responsive information. These media may be mounted in drive or other input/output device and examined to locate the proper files or data structures. Index information may then be located and uploaded to an index or database associated with the appropriate media management component (e.g., media agent). This allows the media management component to locate and retrieve specific data objects on the media that satisfy the search criteria.

Next, if no further media management components have been identified, a list of media management components to be searched may be compiled (step 1945). At step 1950, a list of clients identified as potentially having responsive data objects may also be compiled. After a complete list of secondary storages devices and clients potentially having responsive data objects is identified, the associated metabascs are queried for these components, step 1960, and results are returned indicating data objects that may satisfy the search criteria, step 1965. In some embodiments, these results may be reviewed and analyzed to ensure relevance, with only reasonably relevant or responsive data objects actually being retrieved.

FIG. 20 presents a flow chart 2000 that illustrates some of the steps that may be involved in retrieving data objects from secondary storage (or other tiers or other storage locations) in accordance with principles of the present invention. This may be accomplished generally as follows. Certain data (e.g. data objects or associated metadata) from the system may need to be retrieved. That data may be requested and communicated to the system in the form of a query. The query may be used to search the system and identify media on which responsive data may be located. Once located, data satisfying the selection criteria may be uploaded and retrieved and analyzed for relevance, or other action may be taken. Or, alternatively, the identified data may be moved to other tiers of storage. More specific steps involved in this process may be as follows.

First, at step 2002, a query seeking certain data may be identified. The query may be analyzed to ascertain certain additional information that may assist in identifying responsive information such as identifying a certain point in time to search (step 2004). This may involve consulting storage manager and/or media agent index or database for responsive information relating to a certain point in time. This may also involve consulting certain metabascs for similar information that may be associated with these or other media management components providing copy and management functions. Point in time information may be specified by the user or may be assigned by the system absent a specific time frame established by the user. For example, a user may specify a certain time range within the query (e.g., a time range, a certain date, all information related to a project since its inception etc.).

36

The system however, may assign a certain time limit based on the query (e.g., such as based on the specifics of the query (e.g., only have data relating to a certain time frame)), and may limit the search to the time frame of information present in certain metabascs, master storage manager, or index within the system, and/or poll or otherwise communicate with storage devices within the system to determine the range or time frame of available data within the system and present the user with options for retrieving it (e.g., some, all within a time frame, etc.)

Next, at step 2006 certain archive files may be identified and associated media agents (step 2008) that may have been involved in transferring responsive data objects. This may be determined by consulting a master storage manager or other media management component index or metabase to determine whether the archive files have been handled by such components. Once the appropriate media agents have been identified, it may be determined whether information regarding the identified archive files is present in a cache or index associated with the media agents (step 2010). If not, the index information may need to be uploaded so the appropriate archive files may be retrieved and accessed. This process may be performed until all identified media agents have the appropriate index information loaded and/or until it is determined that no responsive information has been handled by the media agents and therefore no index information need be uploaded.

Next, at step 2016 data objects satisfying the query criteria may be identified by searching metabascs and/or indexes. In some embodiments, such data objects may be compiled into a list of data objects for present or subsequent retrieval. For example, such a list of responsive data objects may be provided to the user (which may itself satisfy the query) and then provide the user with the option to actually retrieve all or certain selected identified data objects.

At step 2018, the new destination for the data objects may be determined. For example, if certain data objects are being migrated off as part of an ILM operation, the query or other information may indicate the intent or reason for the search and the data object's destination. This may be useful in determining whether certain data objects are responsive to search criteria or query. At step 220 it may be determined whether the new destination is primary storage (a restore operation) or secondary or other tier of storage (ILM). Such information may be further useful in determining whether the data objects are likely to fall within a time frame or category of interest and thus may be useful in further identifying data objects of interest.

If the identified data objects are moving to other secondary storage tiers, the data objects may be repackaged into form suitable for secondary storage, which may include repackaging into an archive file, converting to a new format, compressing of the data objects and associated files, encryption, or any other containerization technique known in the art (step 2022).

Once the data objects are in a suitable format, they may be copied to the appropriate storage destination by the system. This may be accomplished by a media agent or media component in conjunction with a storage manager or other media management component that coordinate routing and the specifics involved with file transfer (step 2024), as further described herein. Metadata relating to the copied data objects may then be copied to a metabase associated with a computing device at the destination (step 2026).

For example, metadata relating to the data being copied may be copied along with the data to the secondary storage device and may be copied to an index in the media agent or other media management component involved in the data transfer. This allows the media management component to

US 7,725,671 B2

37

locate and retrieve and otherwise manage the stored data. Such metadata may also be useful when performing searches of secondary storage devices (or other tiers) as further described herein. Metadata stored along with the data on the secondary storage device may be useful to restore or refresh the media agent index in the case of lost or corrupt data and also may be transferred along with the data on storage media in the case whether it is necessary to copy all such data (or actually physically relocate) to another storage device. A master storage manager index or metabase associated with destination computing device may be updated reflecting the arrival and new location of the transferred data objects and/or archive file for system management purposes (step 2034).

In some embodiments, the copied data objects and metadata may be deleted from the source location (steps 2028-2032). For example, at step 2028, it may be determined whether the copied data objects should be deleted based user preferences, storage policy requirements or other system constraints such as diminished storage capacity, etc. At steps 2030 and 2032 the data objects and records may be deleted. However, a stub, pointer or other referential element may be placed at the same logical location to act as a marker for the moved data. This allows subsequent operations to quickly track down and locate the moved data at its new location.

If, however, at step 2020, it is determined that the identified data objects are moving to primary storage, accordingly, the data objects may be reformatted (e.g., unpacked from archive file format) for copying to a computing device (step 2038). Next the unpacked data may be copied to a target computing device along with any associated metadata (steps 2040 and 2042). For example, this may involve reading metadata and/or index information from the archive file and repopulating the metabase and/or management component indexes with this information as further described herein. For example, metadata from the archive file may be retrieved and integrated into a metabase associated with the target computing device including information relating to data management and as well as certain content and storage information as further described herein with respect to the classification process and metabase population. Thus, such archive information may be fully restored to primary storage and any associated information, such as metabase information may be searched and retrieved accordingly.

Moreover, information relating to system management may be uploaded and used to repopulate storage management components within the system such as a storage manager or master storage manager reflecting the return of the retrieved data to primary storage (step 2050). For example, a storage manager index may be updated to reflect the presence of the retrieved data along with certain management information such as logical offsets and location of the retrieved information such that the retrieved information may be located and accessed. Other management components, such as a master storage manager may also be updated with the appropriate identification and location information to reflect the return of the retrieved data within the system.

In certain embodiments, the copied data and metadata may be deleted from the source location (steps 2044-2048). For example, at step 2044, it may be determined whether the copied data objects in secondary storage should be deleted based user preferences, storage policy requirements or other system constraints such as diminished storage capacity, etc. At steps 2046 and 2048 the data objects and records may be deleted within the system including any metabase or other system management information associated with the retrieved data. Storage management components such as stor-

38

age managers, media agents may also be updated to reflect the removal or deletion of such information (step 2050).

Systems and modules described herein may comprise software, firmware, hardware, or any combination(s) of software, firmware, or hardware suitable for the purposes described herein. Software and other modules may reside on servers, workstations, personal computers, computerized tablets, personal digital assistants (PDAs), and other devices suitable for the purposes described herein. In other words, the software and other modules described herein may be executed by a general-purpose computer, e.g., a server computer, wireless device or personal computer. Those skilled in the relevant art will appreciate that aspects of the invention can be practiced with other communications, data processing, or computer system configurations, including: Internet appliances, handheld devices (including PDAs), wearable computers, all manner of cellular or mobile phones, multi-processor systems, microprocessor-based or programmable consumer electronics, set-top boxes, network PCs, mini-computers, mainframe computers, and the like. Indeed, the terms "computer," "server," "host," "host system," and the like are generally used interchangeably herein, and refer to any of the above devices and systems, as well as any data processor. Furthermore, aspects of the invention can be embodied in a special purpose computer or data processor that is specifically programmed, configured, or constructed to perform one or more of the computer-executable instructions explained in detail herein.

Software and other modules may be accessible via local memory, via a network, via a browser or other application in an ASP context, or via other means suitable for the purposes described herein. Aspects of the invention can also be practiced in distributed computing environments where tasks or modules are performed by remote processing devices, which are linked through a communications network, such as a Local Area Network (LAN), Wide Area Network (WAN), or the Internet. In a distributed computing environment, program modules may be located in both local and remote memory storage devices. Data structures described herein may comprise computer files, variables, programming arrays, programming structures, or any electronic information storage schemes or methods, or any combinations thereof, suitable for the purposes described herein. User interface elements described herein may comprise elements from graphical user interfaces, command line interfaces, and other interfaces suitable for the purposes described herein. Screenshots presented and described herein can be displayed differently as known in the art to input, access, change, manipulate, modify, alter, and work with information.

Aspects of the invention may be stored or distributed on computer-readable media, including magnetically or optically readable computer discs, hard-wired or preprogrammed chips (e.g., EEPROM semiconductor chips), nanotechnology memory, biological memory, or other data storage media. Indeed, computer implemented instructions, data structures, screen displays, and other data under aspects of the invention may be distributed over the Internet or over other networks (including wireless networks), on a propagated signal on a propagation medium (e.g., an electromagnetic wave(s), a sound wave, etc.) over a period of time, or they may be provided on any analog or digital network (packet switched, circuit switched, or other scheme).

Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense; that is to say, in the sense of "including, but not limited to." As used herein, the terms "connected," "coupled," or any variant thereof,

US 7,725,671 B2

39

means any connection or coupling, either direct or indirect, between two or more elements; the coupling of connection between the elements can be physical, logical, or a combination thereof. Additionally, the words “herein,” “above,” “below,” and words of similar import, when used in this application, shall refer to this application as a whole and not to any particular portions of this application. Where the context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number respectively. The word “or,” in reference to a list of two or more items, covers all of the following interpretations of the word: any of the items in the list, all of the items in the list, and any combination of the items in the list.

The above detailed description of embodiments of the invention is not intended to be exhaustive or to limit the invention to the precise form disclosed above. While specific embodiments of, and examples for, the invention are described above for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. For example, while processes or blocks are presented in a given order, alternative embodiments may perform routines having steps, or employ systems having blocks, in a different order, and some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified to provide alternative or subcombinations. Each of these processes or blocks may be implemented in a variety of different ways. Also, while processes or blocks are at times shown as being performed in series, these processes or blocks may instead be performed in parallel, or may be performed at different times.

The teachings of the invention provided herein can be applied to other systems, not necessarily the system described above. The elements and acts of the various embodiments described above can be combined to provide further embodiments. Any patents and applications and other references noted above, including any that may be listed in accompanying filing papers, are incorporated herein by reference. Aspects of the invention can be modified, if necessary, to employ the systems, functions, and concepts of the various references described above to provide yet further embodiments of the invention.

We claim:

1. A method of distributing metadata throughout a network, comprising:

receiving an indication to perform a copy operation on one or more data objects;

analyzing a first metabase to identify data objects in a first data store associated with the first metabase that satisfy a selection criteria;

copying the identified data objects from the first data store to a second data store; and

copying metadata associated with the identified data objects from the first metabase to a second metabase, wherein the second metabase is associated with the second data store.

2. The method of claim 1 further comprising, after copying the identified data objects, updating the first metabase with information describing the copied data objects.

3. The method of claim 1 wherein the selection criteria is based on a type of the identified data objects.

4. The method of claim 1 wherein the selection criteria is based on a source of the identified data objects.

5. The method of claim 1 wherein the selection criteria is based on a topology of the network.

40

6. The method of claim 1 wherein the selection criteria is based on information within the metadata.

7. The method of claim 1 wherein sending the selection criteria is based on a migration policy.

8. The method of claim 1 wherein the first data store and first metabase are associated with a first computing device and the second data store and second metabase are associated with a second computing device.

9. The method of claim 1 wherein a computing device accesses data at the second data store when the first data store is unavailable.

10. The method of claim 1 wherein copying metadata includes adding one or more data classifications describing an availability of the copied data at the first data store and the second data store.

11. The method of claim 1 wherein copying metadata includes adding one or more data classifications describing the copy operation.

12. A system of distributing metadata throughout a network, comprising:

a primary data store that contains a production copy of stored data;

a primary metabase component that stores metadata associated with the primary data store, wherein the metadata contains information about data stored at the primary data store and describes characteristics of the data that are not stored with the data;

a secondary data store that contains one or more secondary copies of the data;

a secondary metabase that stores metadata associated with the secondary data store; and

a data management operation component that performs data management operations on the stored data and uses the primary and secondary metabases to facilitate performing data management operations.

13. The system of claim 12 further comprising a metadata migration component that migrates metadata from the primary metabase to the secondary metabase when data is copied from the primary data store to the secondary data store.

14. The system of claim 12 further comprising a data restoration component that copies a subset of the stored data from the secondary data store to the primary data store upon the occurrence of an event, wherein the subset of the stored data is determined using the metadata stored at the primary and secondary metabases.

15. A computer-readable medium encoded with instructions, which when executed by a computer system, perform a method of moving data objects from a primary data store to a secondary data store, the method comprising:

analyzing a primary metabase to determine data objects stored within a primary data store satisfying a data object query;

identifying a secondary data store based on metadata stored within a secondary metabase to receive the determined data objects;

copying the data objects from the primary data store to the secondary data store; and

updating the secondary metabase with information describing the copied data objects.

16. The computer-readable medium of claim 15, wherein the method further comprises, after copying the data objects from the primary data store, deleting the data objects from the primary data store.

US 7,725,671 B2

41

17. The computer-readable medium of claim 15, wherein the method further comprises, after copying the data objects from the primary data store, deleting metadata describing the data objects from the primary metabase.

18. The computer-readable medium of claim 15, wherein the method further comprises, after copying the data objects from the primary data store, updating the primary metabase with information describing the copied data objects.

19. A system for providing redundant access to metadata over a network, comprising:

a means for storing at a centralized primary storage metabase, metadata that contains information about data stored at a computing device and describes characteristics of the data that are not stored with the data;

42

a means for selecting one of multiple distributed secondary storage metabases to store a copy of at least some of the metadata stored at the centralized primary storage metabase; and

a means for storing metadata at the selected one of multiple distributed secondary storage metabases via the network.

20. The system of claim 19 further comprising a means for determining which of the centralized primary storage metabases and the one of multiple distributed secondary storage metabases to use to access the metadata.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 7,725,671 B2
APPLICATION NO. : 11/564194
DATED : May 25, 2010
INVENTOR(S) : Anand Prahlad et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

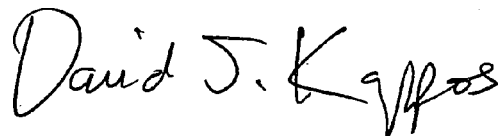
On the Title Page, Item 60, under “Related U.S. Application Data”, line 3-9, after “2005,” delete “provisional application No. 60/752,198, filed on Dec. 19, 2005, provisional application No. 60/752,196, filed on Dec. 19, 2005, provisional application No. 60/752,202, filed on Dec. 19, 2005, provisional application No. 60/752,201, filed on Dec. 19, 2005, provisional application No. 60/752,197, filed on Dec. 19, 2005.”.

In column 10, line 5, delete “(primary)” and insert -- (primary, --, therefor.

In column 40, line 46, in claim 14, delete “occurence” and insert -- occurrence --, therefor.

Signed and Sealed this

Twenty-fourth Day of August, 2010

A handwritten signature in black ink, reading "David J. Kappos". The signature is written in a cursive, flowing style with a large initial 'D' and 'K'.

David J. Kappos
Director of the United States Patent and Trademark Office

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

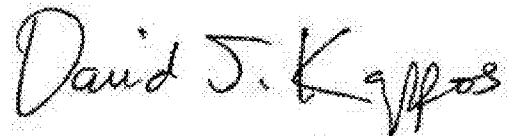
PATENT NO. : 7,725,671 B2
APPLICATION NO. : 11/564194
DATED : May 25, 2010
INVENTOR(S) : Anand Prahlad et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page, Item (73), Assignee: delete "Comm Vault" and insert -- CommVault --, therefor.

Signed and Sealed this
Fifth Day of July, 2011

A handwritten signature in black ink that reads "David J. Kappos". The signature is written in a cursive, flowing style with a large initial "D" and a stylized "K".

David J. Kappos
Director of the United States Patent and Trademark Office

EXHIBIT B

(12) **United States Patent**
Prahlad et al.

(10) **Patent No.:** **US 7,840,533 B2**
(45) **Date of Patent:** ***Nov. 23, 2010**

(54) **SYSTEM AND METHOD FOR PERFORMING AN IMAGE LEVEL SNAPSHOT AND FOR RESTORING PARTIAL VOLUME DATA**

(75) Inventors: **Anand Prahlad**, East Brunswick, NJ (US); **David Ngo**, Shrewsbury, NJ (US); **Prakash Varadharajan**, Oldbridge, NJ (US); **Rahual Pawar**, Ocean, NJ (US); **Avinash Kumar**, Ocean, NJ (US)

(73) Assignee: **CommVault Systems, Inc.**, Oceanport, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 58 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **12/433,238**

(22) Filed: **Apr. 30, 2009**

(65) **Prior Publication Data**

US 2009/0240748 A1 Sep. 24, 2009

Related U.S. Application Data

(63) Continuation of application No. 10/990,353, filed on Nov. 15, 2004, now Pat. No. 7,539,707.

(60) Provisional application No. 60/519,876, filed on Nov. 13, 2003, provisional application No. 60/519,576, filed on Nov. 13, 2003.

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.** **707/610; 707/639**

(58) **Field of Classification Search** **707/609, 707/610, 639**

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,686,620 A	8/1987	Ng
4,995,035 A	2/1991	Cole et al.
5,005,122 A	4/1991	Griffin et al.
5,093,912 A	3/1992	Dong et al.

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0259912	3/1988
----	---------	--------

(Continued)

OTHER PUBLICATIONS

Jason Gait, "The Optical File Cabinet: A Random-Access File System for Write-Once Optical Disks," *IEEE Computer*, vol. 21, No. 6, pp. 11-22 (1988) (see in particular figure 5 in p. 15 and recitation in claim 5).

(Continued)

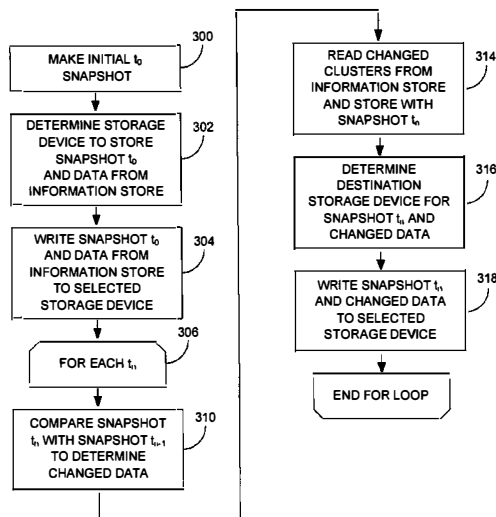
Primary Examiner—Jean M Corrielus

(74) *Attorney, Agent, or Firm*—Perkins Coie LLP

(57) **ABSTRACT**

The present invention relates to a method for performing an image level copy of an information store. The present invention comprises performing a snapshot of an information store that indexes the contents of the information store, retrieving data associated with the contents of the information store from a file allocation table, copying the contents of the information store to a storage device based on the snapshot, and associating the retrieved data with the copied contents to provide file system information for the copied contents.

14 Claims, 6 Drawing Sheets



US 7,840,533 B2

Page 2

U.S. PATENT DOCUMENTS

5,133,065	A	7/1992	Cheffetz et al.	6,275,953	B1	8/2001	Vahalia et al.
5,193,154	A	3/1993	Kitajima et al.	6,301,592	B1	10/2001	Aoyama et al.
5,212,772	A	5/1993	Masters	6,311,193	B1	10/2001	Sekido
5,226,157	A	7/1993	Nakano et al.	6,324,581	B1	11/2001	Xu et al.
5,239,647	A	8/1993	Anglin et al.	6,328,766	B1	12/2001	Long
5,241,668	A	8/1993	Eastridge et al.	6,330,570	B1	12/2001	Crighton et al.
5,241,670	A	8/1993	Eastridge et al.	6,330,642	B1	12/2001	Carteau
5,263,154	A	11/1993	Eastridge et al.	6,343,324	B1	1/2002	Hubis et al.
5,276,860	A	1/1994	Fortier et al.	RE37,601	E	3/2002	Eastridge et al.
5,276,867	A	1/1994	Kenley et al.	6,356,801	B1	3/2002	Goodman et al.
5,287,500	A	2/1994	Stoppini, Jr.	6,366,986	B1	4/2002	St. Pierre et al.
5,317,731	A	5/1994	Dias et al.	6,366,988	B1	4/2002	Skiba et al.
5,321,816	A	6/1994	Rogan et al.	6,374,363	B1	4/2002	Wu et al.
5,333,315	A	7/1994	Saether et al.	6,389,432	B1	5/2002	Pothapragada et al.
5,347,653	A	9/1994	Flynn et al.	6,418,478	B1	7/2002	Ignatius et al.
5,369,757	A	11/1994	Spiro et al.	6,421,711	B1	7/2002	Blumenau et al.
5,403,639	A	4/1995	Belsan et al.	6,434,681	B1	8/2002	Armangau
5,410,700	A	4/1995	Fecteau et al.	6,473,775	B1	10/2002	Kusters et al.
5,448,724	A	9/1995	Hayashi et al.	6,487,561	B1	11/2002	●fek et al.
5,491,810	A	2/1996	Allen	6,519,679	B2	2/2003	Devireddy et al.
5,495,607	A	2/1996	Pisello et al.	6,538,669	B1	3/2003	Lagueux, Jr. et al.
5,504,873	A	4/1996	Martin et al.	6,564,228	B1	5/2003	●'Connor
5,544,345	A	8/1996	Carpenter et al.	6,604,118	B2	8/2003	Kleiman et al.
5,544,347	A	8/1996	Yanai et al.	6,631,477	B1	10/2003	LeCrone et al.
5,559,957	A	9/1996	Balk	6,643,671	B2	11/2003	Milillo et al.
5,559,991	A	9/1996	Kanfi	6,647,473	B1	11/2003	Golds et al.
5,604,862	A	2/1997	Midgely et al.	6,651,075	B1	11/2003	Kusters et al.
5,619,644	A	4/1997	Crockett et al.	6,658,526	B2	12/2003	Nguyen et al.
5,638,509	A	6/1997	Dunphy et al.	6,662,198	B2	12/2003	Satyanarayanan et al.
5,642,496	A	6/1997	Kanfi	6,665,815	B1	12/2003	Goldstein et al.
5,673,381	A	9/1997	Huai et al.	6,721,767	B2	4/2004	De Meno et al.
5,699,361	A	12/1997	Ding et al.	6,728,736	B2	4/2004	Hostetter et al.
5,720,026	A	2/1998	Uemura et al.	6,732,125	B1	5/2004	Autrey et al.
5,729,743	A	3/1998	Squibb	6,760,723	B2	7/2004	●shinsky et al.
5,751,997	A	5/1998	Kullick et al.	6,792,518	B2	9/2004	Armangau et al.
5,758,359	A	5/1998	Saxon	6,799,258	B1	9/2004	Linde
5,761,677	A	6/1998	Senator et al.	6,871,271	B2	3/2005	●hran et al.
5,764,972	A	6/1998	Crouse et al.	6,880,051	B2	4/2005	Timpanaro-Perrotta
5,765,173	A	6/1998	Cane et al.	6,898,688	B2	5/2005	Martin et al.
5,778,395	A	7/1998	Whiting et al.	6,938,135	B1	8/2005	Kekre et al.
5,790,114	A	8/1998	Geaghan et al.	6,948,038	B2	9/2005	Berkowitz et al.
5,812,398	A	9/1998	Nielsen	6,948,089	B2	9/2005	Fujibayashi
5,813,009	A	9/1998	Johnson et al.	6,954,834	B2	10/2005	Slater et al.
5,813,017	A	9/1998	Morris	6,981,177	B2	12/2005	Beattie
5,875,478	A	2/1999	Blumenau	6,993,539	B2	1/2006	Federwisch et al.
5,878,408	A	3/1999	Van Huben et al.	7,003,641	B2	2/2006	Prahlad et al.
5,887,134	A	3/1999	Ebrahim	7,035,880	B1	4/2006	Crescenti et al.
5,901,327	A	5/1999	●fek	7,165,079	B1 *	1/2007	Chen et al. 1/1
5,907,672	A	5/1999	Matze et al.	7,174,352	B2	2/2007	Kleiman et al.
5,924,102	A	7/1999	Perks	7,209,972	B1	4/2007	Ignatius et al.
5,950,205	A	9/1999	Aviani, Jr.	7,225,204	B2	5/2007	Manley et al.
5,974,563	A	10/1999	Beeler, Jr.	7,225,208	B2	5/2007	Midgley et al.
6,021,415	A	2/2000	Cannon et al.	7,225,210	B2	5/2007	Guthrie, II
6,021,475	A	2/2000	Nguyen et al.	7,231,544	B2	6/2007	Tan et al.
6,026,414	A	2/2000	Anglin	7,234,115	B1	6/2007	Sprauve et al.
6,052,735	A	4/2000	Ulrich et al.	7,237,075	B2	6/2007	Welsh et al.
6,072,490	A	6/2000	Bates et al.	7,275,177	B2 *	9/2007	Armangau et al. 714/5
6,076,148	A	6/2000	Kedem et al.	7,296,125	B2	11/2007	●hran
6,094,416	A	7/2000	Ying	7,346,623	B2	3/2008	Prahlad et al.
6,131,095	A	10/2000	Low et al.	7,383,538	B2	6/2008	Bates et al.
6,131,148	A	10/2000	West et al.	7,395,282	B1	7/2008	Crescenti et al.
6,131,190	A	10/2000	Sidwell	7,412,583	B2	8/2008	Burton et al.
6,148,412	A	11/2000	Cannon et al.	7,529,782	B2 *	5/2009	Prahlad et al. 1/1
6,154,787	A	11/2000	Urevig et al.	7,539,707	B2 *	5/2009	Prahlad et al. 1/1
6,161,111	A	12/2000	Mutalik et al.	7,567,991	B2 *	7/2009	Armangau et al. 1/1
6,167,402	A	12/2000	Yeager	7,620,666	B1 *	11/2009	Root et al. 1/1
6,195,695	B1	2/2001	Cheston et al.	7,734,578	B2 *	6/2010	Prahlad et al. 707/609
6,205,450	B1	3/2001	Kanome	2002/0107877	A1	8/2002	Whiting et al.
6,212,512	B1	4/2001	Barney et al.	2003/0167380	A1	9/2003	Green et al.
6,260,069	B1	7/2001	Anglin	2003/0177149	A1	9/2003	Coombs
6,269,431	B1	7/2001	Dunham	2004/0139128	A1	7/2004	Becker et al.
				2004/0236958	A1	11/2004	Teicher et al.
				2004/0250033	A1	12/2004	Prahlad et al.

US 7,840,533 B2

Page 3

2004/0260678 A1 12/2004 Verbowski et al.
 2004/0267836 A1 12/2004 Armangau et al.
 2007/0185940 A1 8/2007 Prahlad et al.
 2008/0183775 A1 7/2008 Prahlad et al.

FOREIGN PATENT DOCUMENTS

EP	0405926	1/1991
EP	0467546	1/1992
EP	0774715	5/1997
EP	0809184	11/1997
EP	0899662	3/1999
EP	0981090	2/2000
EP	1349088	10/2003
EP	1579331	9/2005
GB	2256952	12/1992
GB	2411030	8/2005
JP	05189281	7/1993
JP	06274605	9/1994
JP	09016463	1/1997
JP	11259348	9/1999
JP	2000347811	12/2000
W●	W●-9303549	2/1993
W●	W●-95/13580	5/1995
W●	W●-99/12098	3/1999
W●	W●-03028183	4/2003
W●	W●-2004034197	4/2004

OTHER PUBLICATIONS

Rosenblum et al., "The Design and Implementation of a Log-Structured File System," *Operating Systems Review SIGOPS*, vol. 25, No. 5, New York, US, pp. 1-15 (May 1991).

Arneson, "Mass Storage Archiving in Network Environments," Digest of Papers, Ninth IEEE Symposium on Mass Storage Systems, Oct. 31, 1988-Jan. 3, 1988, pp. 45-50, Monterey, CA.

Eitel, "Backup and Storage Management in Distributed Heterogeneous Environments," *IEEE*, 1994, pp. 124-126.

Jander, M., "Launching Storage-Area Net," *Data Communications*, US, McGraw Hill, NY, vol. 27, No. 4 (Mar. 21, 1998), pp. 64-72.

Cabrera et al., "ADSM: A Multi-Platform, Scalable, Backup and Archive Mass Storage System," Digest of Papers, Compcon '95, Proceedings of the 40th IEEE Computer Society International Conference, Mar. 5, 1995-Mar. 9, 1995, pp. 420-427, San Francisco, CA.

Armstead et al., "Implementation of a Campus-wide Distributed Mass Storage Service: The Dream vs. Reality," *IEEE*, 1995, pp. 190-199.

Non-Final Office Action for U.S. Appl. No. 10/990,353, Mail Date Sep. 15, 2008, 8 pages.

Non-Final Office Action for U.S. Appl. No. 10/681,386, Mail Date Oct. 28, 2008, 16 pages.

Non-Final Office Action for U.S. Appl. No. 11/672,926, Mail Date Nov. 25, 2008, 21 pages.

Examiner's Report for Australian Application No. 2003279847, Dated Dec. 9, 2008, 4 pages.

Notice of Allowance for U.S. Appl. No. 10/990,353, Mail Date Apr. 7, 2009, 11 pages.

Notice of Allowance for U.S. Appl. No. 10/681,386, Mail Date Apr. 21, 2009, 10 pages.

First Office Action for Japanese Application No. 2003-531581, Mail Date Jul. 8 2008, 8 pages.

Final Office Action for Japanese Application No. 2003-531581, Mail Date Mar. 24, 2009, 6 pages.

Veritas Software Corporation, "Veritas Volume Manager 3.2, Administrator's Guide," Aug. 2001, 360 pages.

* cited by examiner

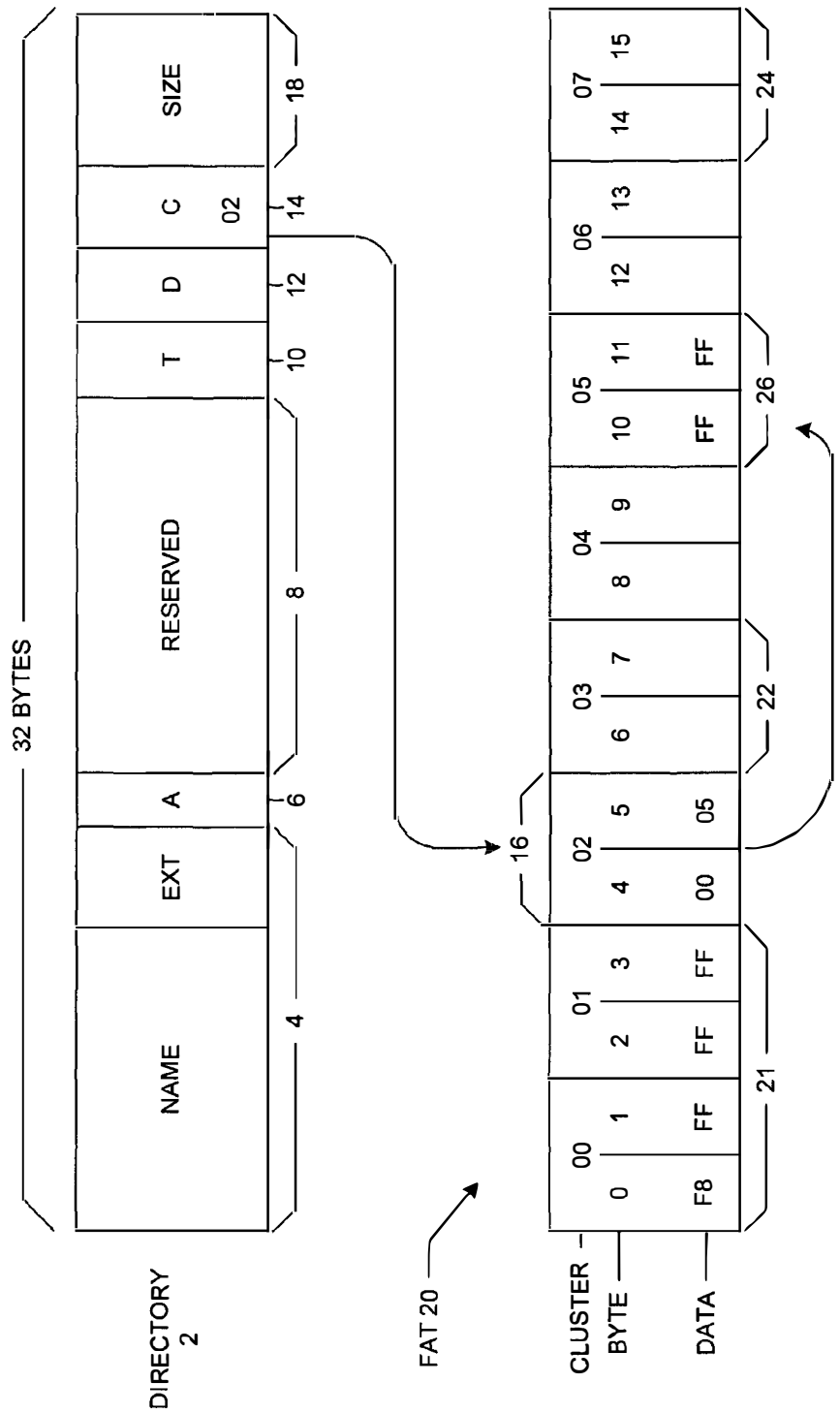


FIG. 1
(Prior Art)

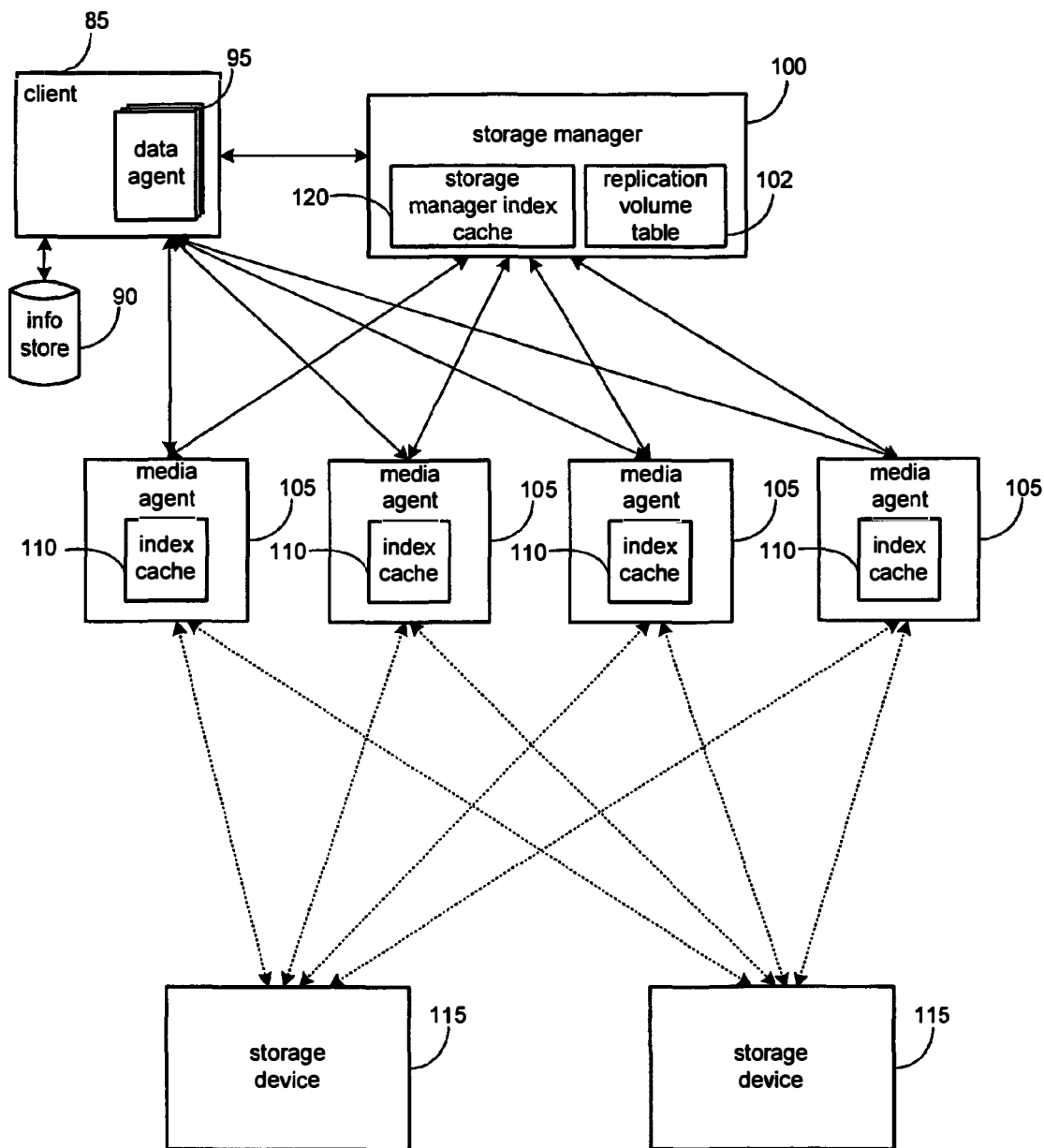


Fig. 2

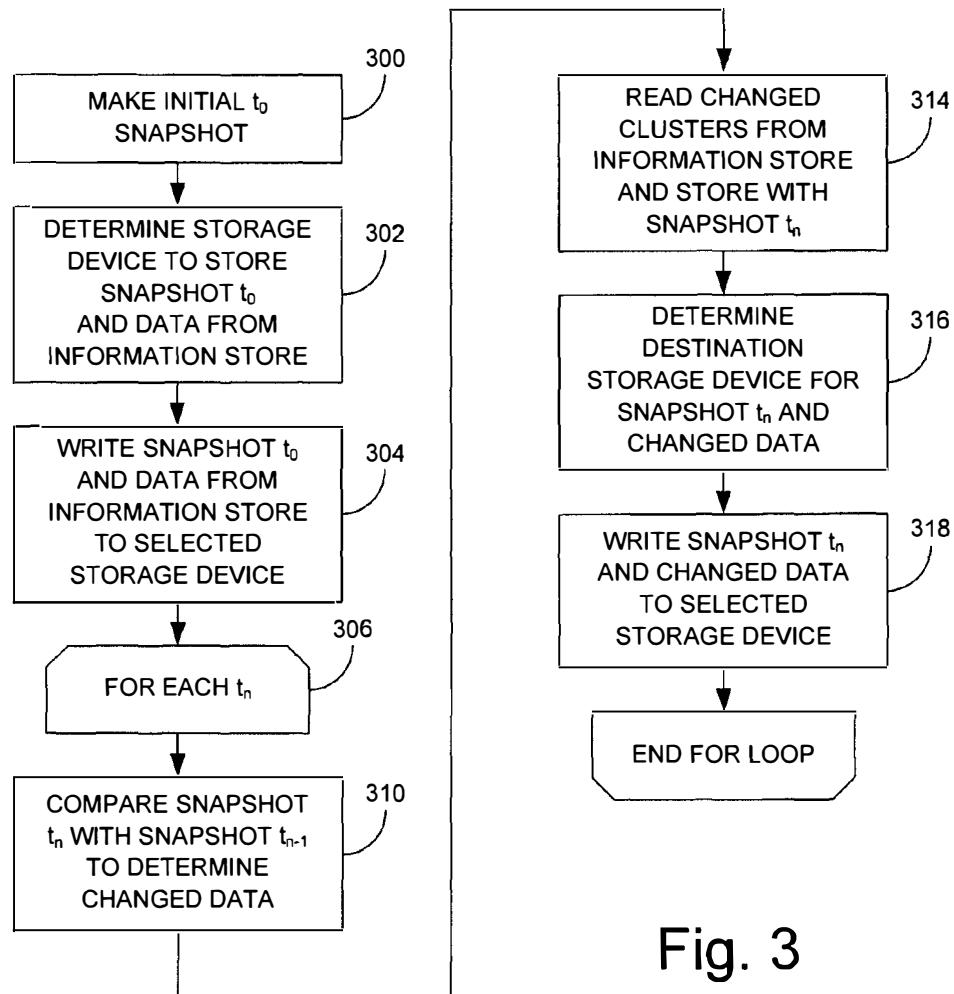


Fig. 3

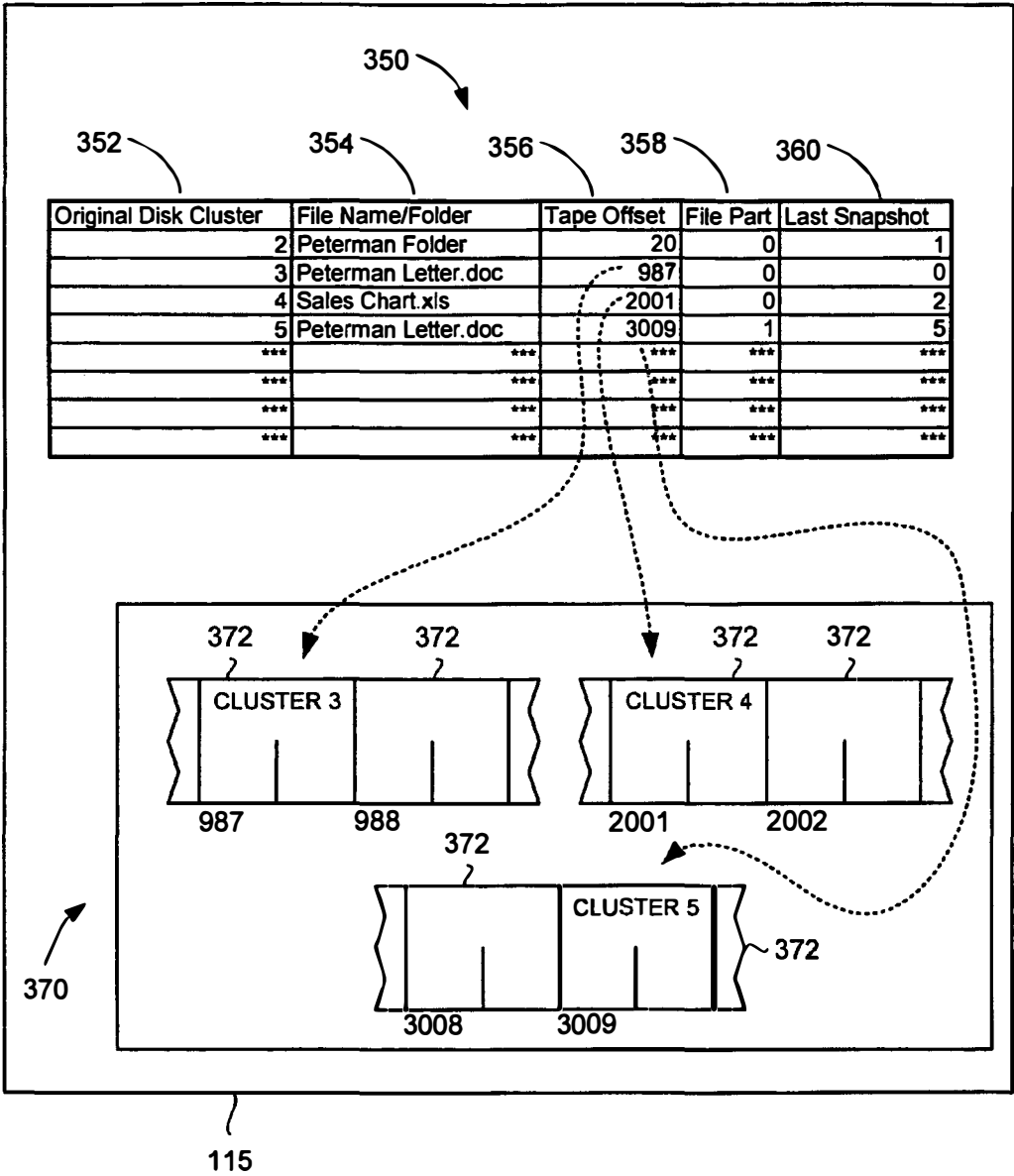


Fig. 4

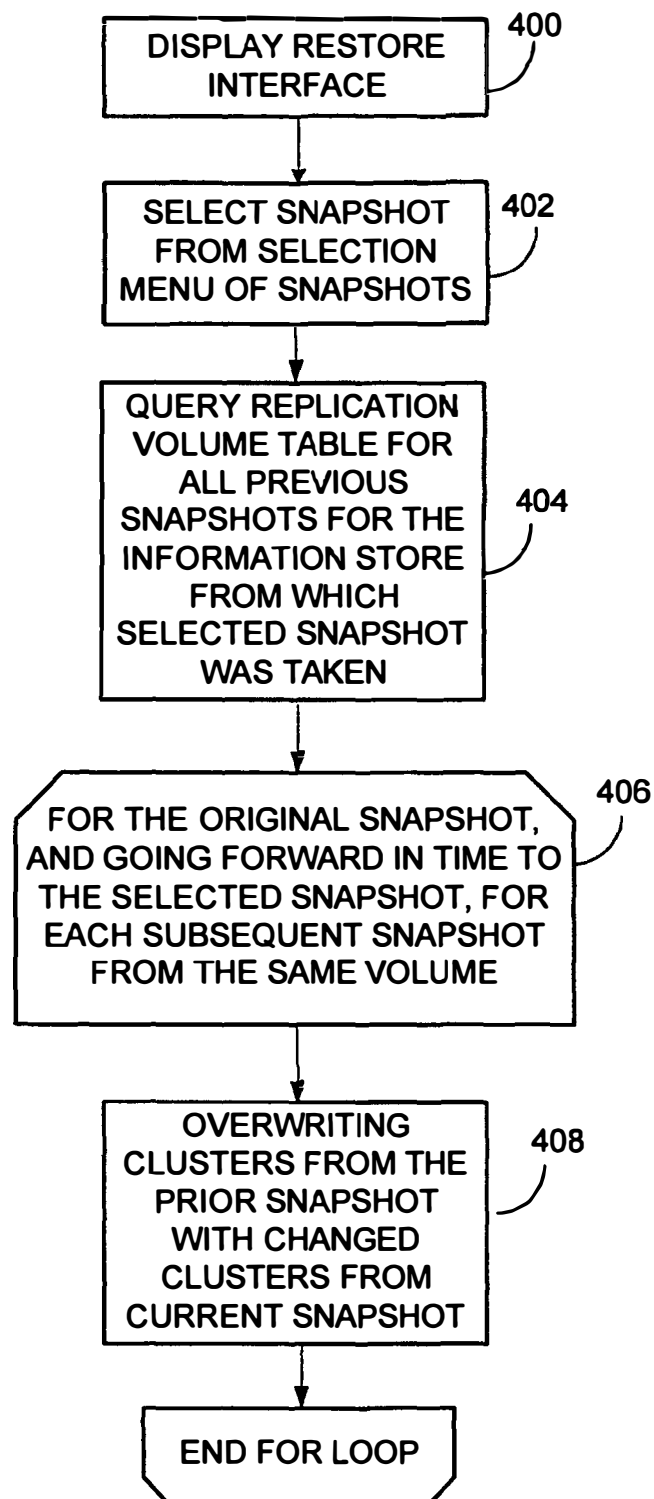


Fig. 5

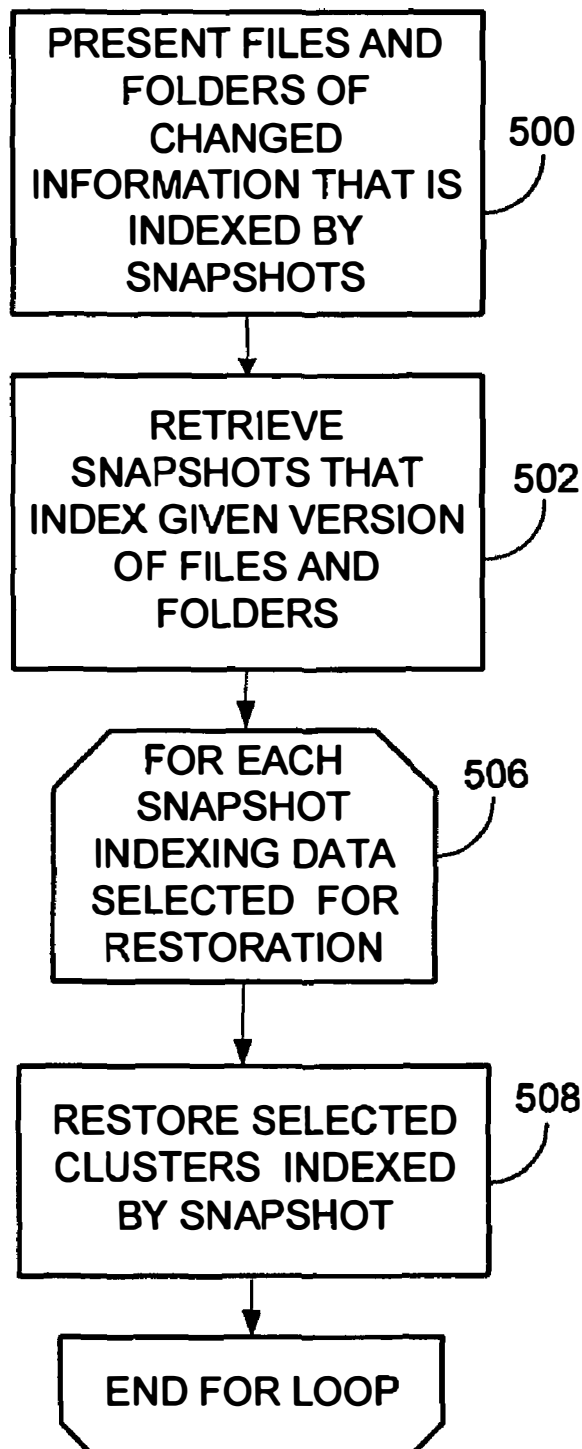


Fig. 6

US 7,840,533 B2

1

SYSTEM AND METHOD FOR PERFORMING AN IMAGE LEVEL SNAPSHOT AND FOR RESTORING PARTIAL VOLUME DATA

This is a continuation of U.S. application Ser. No. 10/990, 353, filed on Nov. 15, 2004 now U.S. Pat. No. 7,539,707, which is herein incorporated by reference in its entirety, and which claims the benefit of provisional patent application Ser. Nos. 60/519,876 and 60/519,576, entitled "SYSTEM AND METHOD FOR PERFORMING A SNAPSHOT AND FOR RESTORING DATA," and "SYSTEM AND METHOD FOR PERFORMING AN IMAGE LEVEL SNAPSHOT AND FOR RESTORING PARTIAL VOLUME DATA," respectively, each filed on Nov. 13, 2003. These applications are incorporated by reference herein in their entirety.

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosures, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

RELATED APPLICATIONS

This application is related to the following patents and pending patent applications, each of which is hereby incorporated herein by reference in its entirety:

U.S. Pat. No. 6,418,478, entitled "PIPELINED HIGH SPEED DATA TRANSFER MECHANISM," issued Jul. 9, 2002;

application Ser. No. 09/610,738, entitled "MODULAR BACKUP AND RETRIEVAL SYSTEM USED IN CONJUNCTION WITH A STORAGE AREA NETWORK," filed Jul. 6, 2000;

application Ser. No. 09/744,268, entitled "LOGICAL VIEW AND ACCESS TO PHYSICAL STORAGE IN MODULAR DATA AND STORAGE MANAGEMENT SYSTEM," filed Jan. 30, 2001;

Application Ser. No. 60/409,183, entitled "DYNAMIC STORAGE DEVICE POOLING IN A COMPUTER SYSTEM," filed Sep. 9, 2002;

application Ser. No. 10/681,386 entitled "SYSTEM AND METHOD FOR MANAGING STORED DATA," filed Oct. 7, 2003; and

Application Ser. No. 60/460,234, entitled "SYSTEM AND METHOD FOR PERFORMING STORAGE OPERATIONS IN A COMPUTER NETWORK," filed Apr. 3, 2003.

BACKGROUND OF THE INVENTION

The invention disclosed herein relates generally to a system and method for performing a snapshot and for restoring data. More particularly, the present invention relates to a system and method for performing snapshots of an information store, which are stored across multiple storage devices, and for restoring partial or full snapshots.

To obtain a more thorough understanding of the present invention, the following discussion provides additional understanding regarding the manner in which magnetic media is used to store information. Using traditional techniques, copies of an information store are performed using the operating system's file system. Copying is done by accessing the

2

operating system's (OS) file system for the information store to be backed-up, such as the Windows NTFS file system. The file allocation system of the operating system typically uses a file allocation table to keep track of the physical or logical clusters across which each file in the information store is stored. Also called an allocation unit, a cluster is a given number of disk sectors that are treated as a unit, each disk sector storing a number of bytes of data. This unit, the cluster, is the smallest unit of storage the operating system can manage. For example, on a computer running Microsoft's Windows 95 operating system, the OS uses the Windows FAT32 32-bit file allocation table having a cluster size to 4K. The number of sectors is determined when the disk is formatted by a formatting program, generally, but not necessarily, when the OS is installed.

The operating system allocates disk space for a file only when needed. That is, the data space is not preallocated but allocated dynamically. The space is allocated one cluster at a time, where a cluster is a given number of consecutive disk sectors. The clusters for a file are chained together, and kept track of, by entries in a file allocation table (FAT).

The clusters are arranged on the disk to minimize the disk head movement. For example, all of the space on a track is allocated before moving on to the next track. This is accomplished by using the sequential sectors on the lowest-numbered cylinder of the lowest numbered platter, then all sectors in the cylinder on the next platter, and so on, until all sectors on all platters of the cylinder are used. This is performed sequentially across the entire disk, for example, the next sector to be used will be sector 1 on platter 0 of the next cylinder.

For a hard (fixed) disk, FAT, sector, cluster, etc. size is determined when a disk formatting program formats the disk, and are based on the size of the partition. To locate all of the data that is associated with a particular file stored on a hard disk, the starting cluster of the file is obtained from the directory entry, then the FAT is referenced to locate the next cluster associated with the file. Essentially, the FAT is a linked list of pointers to clusters on the disk, e.g., each 16-bit FAT entry for a file points to the next sequential cluster used for that file. The last entry for a file in the FAT has a number indicating that no more clusters follow. This number can be from FFF8 to FFFF (base 16) inclusive.

FIG. 1 shows an example directory entry 2 of a Windows-formatted hard disk and accompanying FAT 20. The exemplary directory entry 2 consists of 32 bytes of data. The name of the file and its extension are stored in the first eleven bytes 4 of the directory entry 2 and a file attribute byte 6 is provided. By definition, ten bytes 8 are reserved for future use and four bytes are provided to store time 10 and date 12 information (two bytes each). Two cluster bytes 14 point to the first cluster of sectors used to store the file information. The last four bytes 18 of the directory entry 2 are used to store the size of the file.

A sixteen-byte section of a FAT 20 is depicted. The first four bytes 21 store system information. A two-byte pair, bytes four and five (16), are the beginning bytes of the FAT 20 used to track file information. The first cluster for data space on all disks is cluster "02." Therefore, bytes four and five (16) are associated with the first cluster of disk sectors "02" used to store file information. Bytes six and seven (22) are associated with cluster "03" . . . and bytes fourteen and fifteen (24) are associated with cluster "07."

This example illustrates how sectors associated with a file referenced in a directory are located. The cluster information bytes 14 in the directory 2 point to cluster number "02." The sectors in cluster "02" (not shown), contain the initial sector of data for the referenced file. Next, the FAT is referenced to

US 7,840,533 B2

3

see if additional clusters are used to store the file information. FAT bytes four and five (16) were pointed to by the cluster information bytes 14, and the information stored in bytes four and five (16) in the FAT 20 point to the next cluster used for the file. Here, the next cluster is "05". Accordingly, cluster "05" contains the next sector of data for the referenced file. FAT bytes ten and eleven (26) contain an end-of-file flag, "FFFF," indicating there are no more clusters associated with the referenced file. All of the information comprising the referenced file, therefore, is contained in clusters "02" and "05" on the disk.

As with other applications running on the computer, a typical backup application provides a read request to the operating system, which handles interpretation of the information contained in the FAT and reading of each file for the backup application. A file system is provided on the storage device that is used by the backup application to write files that are copied to the device. Similarly, the recovery portion of the backup application, or a separate recovery application, may read files from the storage device for recovery of the information.

Inherent problems and disadvantages have been discovered with currently available systems and methods for archiving data contained in an information store. One technique is to perform a full copy of the data contained in the information store. Utilizing this technique results in two separate copies of the information store, and the length of time it takes to make this kind of copy is related to the amount of data copied and the speed of the disk subsystem. For example, assuming a transfer rate of 25 MB/sec, the approach will take one hour to copy 90 GB of data. These techniques, however, in addition to other disadvantages, require the applications on the information store to be quiesced during the copy routine. This places a significant burden on system administrators to complete copying and get critical systems back into the production environment as, absent a high-speed data bus, the copying may consume a significant amount of time to complete.

Administrators typically keep multiple copies of a given information store. Unfortunately, this has the drawback of requiring n times the amount of space of the information store to maintain n copies, which can be quite expensive to store, in addition to requiring complex and time consuming techniques for restoration of the copied data.

One currently available alternative is to perform snapshots of an information store. With current snapshot systems and methods, administrators create an incremental copy that is an exact point-in-time replica of the source volume each time a snapshot is taken. A series of snapshot are stored locally on the information store from which it was taken and track incremental changes to the data in the information store. Furthermore, changed data is written to a new location in the information store as tracked by the snapshot. With knowledge regarding the change, as well as the changed data, the snapshot can be used to "roll back" changes to an information store to the point in time when the snapshot was taken. If there should be any logical corruption in the information store's data that went un-detected for a period of time, however, these incremental updates faithfully replicate that logical corruption to the data when copying. Additionally, other drawbacks are associated with currently know snapshot techniques, including the significant drawback of preventing restoration from the snapshot in the event that the information store fails, as both the snapshot and the information store become unavailable.

4

Systems and methods are needed, therefore, that overcome problems associated with currently known techniques for taking, maintaining and restoring snapshots.

SUMMARY OF THE INVENTION

The present invention addresses, among other things, the problems discussed above with copying up data using systems and methods known to those of skill in the art. The invention provides systems and methods for performing n snapshots of an information store, without requiring n times the space of the information store, and storing those snapshots in multiple destinations across a network.

One embodiment of the system of the present invention creates the snapshots by taking a snapshot that indexes only clusters for files that were created or changed since the last snapshot. A snapshots, t_n , is restored by restoring the clusters from the snapshot t_n . The clusters that were not restored from snapshot t_n are restored from snapshot t_{n-1} , etc., until the remaining clusters are restored from the first snapshot, snapshot t_0 .

In accordance with some aspects of the present invention, multiple snapshots are kept on a storage device, without requiring n times the space of the total volume of the information store. The system creates snapshots at various points in time that index only clusters for files that were created or changed since the last snapshot, and creates a copy of the data that has been changed or created. This allows users to keep several snapshots without requiring n times the space of the total volume of the information store.

In some embodiments, the system stores a map, which may be part of a snapshot, to track specific files and folders with their corresponding copied clusters. The map created by reading data from the file allocation table of the information store and associates files and folders with the clusters stored in the snapshots. In this way, even though the snapshot was performed at the cluster level, individual or groups of files and/or folders may be restored without unnecessarily restoring the entire information store.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is illustrated in the figures of the accompanying drawings which are meant to be exemplary and not limiting, in which like references are intended to refer to like or corresponding parts, and in which:

FIG. 1 is an example directory entry for a file in a prior art FAT of a Windows-formatted hard disk;

FIG. 2 is a block diagram illustrating a network architecture for performing snapshot operations according to one embodiment of the present invention;

FIG. 3 is a flow diagram illustrating a method for creating a snapshot according to one embodiment of the present invention;

FIG. 4 is a block diagram illustrating the relationship between a map and a snapshot according to one embodiment of the present invention;

FIG. 5 is a flow diagram illustrating a method for restoring a snapshot according to one embodiment of the present invention; and

US 7,840,533 B2

5

FIG. 6 is a flow diagram illustrating a method for restoring specific files or folders from a snapshot according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

With reference to FIGS. 2 through 6, embodiments of the present invention are shown. FIG. 2 presents a block diagram illustrating the components of a system for performing storage and restoration operations on electronic data in a computer network according to one embodiment of the invention. It should be understood that the invention is not limited to networked environments, and may also be implemented on a stand-alone computer or electronic device.

As shown, the system of FIG. 2 includes a storage manager 100, including a volume replication table 102 and a storage manager index cache 120, and one or more of the following: a client 85, an information store 90, a data agent 95, a media agent 105, a media agent index cache 110, and a storage device 115. One exemplary embodiment of the present system is the CommVault QuNetix three-tier system available from CommVault Systems, Inc. of Oceanport, N.J., further described in U.S. patent application Ser. No. 09/610,738 and hereby incorporated by reference in its entirety.

A data agent 95 is a software module that is generally responsible for retrieving data from an information store 90 for copies, snapshots, archiving, migration, and recovery of data stored in an information store 90 or other memory location, e.g., hard disc drive. Each client computer 85 preferably has at least one data agent 95 and the system can support many client computers 85. The data agent 95 provides an interface to an information store 90 to execute copies, snapshots, archiving, migration, recovery and other storage operations on data in conjunction with one or more media agents 105. According to one embodiment, each client 85 runs a number of data agents 95, wherein each data agent is configured to interface with data generated by or from a specific application, e.g., a first data agent to interface with Microsoft Exchange data and a second data agent to interface with Oracle database data. As is explained in greater detail herein, a data agent 95 is in communication with one or more media agents 105 to effect the distributed storage of snapshots on one or more storage devices 115 that are remote from the information store that is the source of the snapshot 90.

The storage manager 100 is a software module or application that coordinates and controls other components comprising the system, e.g., data and media agents, 95 and 105, respectively. The storage manager 100 communicates with data 95 and media 105 agents to control and manage snapshot creation, migration, recovery and other storage operations. According to one embodiment, the storage manager 100 maintains data in a storage manager index cache 120 that instructs a given data agent 95 to work in conjunction with a specific media agent 105 to store snapshots on one or more storage devices 115.

The storage manager 100 maintains a storage manager index cache 120. Data in the storage manager index cache 120, which the storage manager 100 collects from data agents 95, media agents 105, user and other applications, is used to indicate, track and associate: logical relationships and associations between components of the system, user preferences, management tasks, and other data that is useful to the system. For example, the storage manager index cache 120 may contain data that tracks logical associations between media agents 105 and storage devices 115. The storage manager index cache 120 may also contain data that tracks the status of

6

storage operations to be performed, storage patterns such as media use, storage space growth, network bandwidth, service level agreement ("SLA") compliance levels, data protection levels, storage policy information, storage criteria associated with user preferences, data retention criteria, storage operation preferences, and other storage-related information.

A media agent 105 is a software module that transfers data in conjunction with one or more data agents 95, as directed by the storage manager 100, between an information store 90 and one or more storage devices 115, such as a tape library, a magnetic media storage device, an optical media storage device, or other storage device. The media agent 105 communicates with and controls the one or more storage devices 115. According to one embodiment, the media agent 105 may communicate with the storage device 115 via a local bus, such as a SCSI adaptor. Alternatively, the storage device 115 may communicate with the data agent 105 via a Storage Area Network ("SAN"). Other types of communication techniques, protocols and media are contemplated as falling within the scope of the invention.

The media agent 105 receives snapshots, preferably with the changed data that is tracked by the snapshot, from one or more data agents 95 and determines one or more storage devices 115 to which it should write the snapshot. According to one embodiment, the media agent 105 applies load-balancing algorithms to select a storage device 115 to which it writes the snapshot. Alternatively, the storage manager 100 instructs the media agent 105 as to which storage device 115 the snapshot should be written. In this manner, snapshots from a given information store 90 may be written to one or more storage devices 115, ensuring data is available for restoration purposes in the event that the information store fails. Either the media agent or the storage manager 100 records the storage device on which the snapshot is written in a replication volume table 102, thereby allowing the snapshot to be located when required for restoring the information store 90.

A media agent 105 maintains a media agent index cache 110 that stores index data the system generates during snapshot, migration, and restore operations. For example, storage operations for Microsoft Exchange data generate application specific index data regarding the substantive Exchange data. Similarly, other applications may be capable of generating application specific data during a copy or snapshot. This data is generally described as metadata, and may be stored in the media agent index cache 110. The media agent index cache 110 may track data that includes, for example, information regarding the location of stored data on a given volume. The media agent index cache 110 may also track data that includes, but is not limited to, file names, sizes, creation dates, formats, application types, and other file-related information, information regarding one or more clients associated stored data, information regarding one or more storage policies, storage criteria, storage preferences, compression information, retention-related information, encryption related information, and stream related information. Index data provides the system with an efficient mechanism for locating user files during storage operations such as copying, performing snapshots and recovery.

This index data is preferably stored with the snapshot that is backed up to the storage device 115, although it is not required, and the media agent 105 that controls the storage operation may also write an additional copy of the index data to its media agent index cache 110. The data in the media agent index cache 110 is thus readily available to the system for use in storage operations and other activities without having to be first retrieved from the storage device 115.

7

In order to track the location of snapshots, the system uses a database table or similar data structure, referred to herein as a replication volume table **102**. The replication volume table **102**, among other advantages, facilitates the tracking of multiple snapshots across multiple storage devices **115**. For example, the system might, as directed by a policy or a user, store a first snapshot t_0 on first storage device A, such as a tape drive or library, and then store subsequent snapshots containing only the changed culster(s), t_n , on a second storage device B, such as an optical drive or library. Alternatively, instructions may be stored within system components, e.g., a storage manger **100** or media agent **105**, directing the storage devices **115** used to store snapshots. Information regarding the storage device **115** to which the snapshot is written, as well as other information regarding the snapshot generally, is written to the replication volume table **102**. An exemplary structure according to one embodiment is as follows:

id	serial,	// PRIMARY KEY FOR THIS TABLE
PointInTime	integer,	//
CreationTime	integer,	// Timestamp of RV creation
ModifyTime	integer,	// Timestamp of last RV update
Current State	integer,	// Current state of RV
CurrentRole	integer,	// Current role of RV
PrimaryVolumeId	integer,	// FOREIGN KEY FOR SNRVOLUME TABLE
PhysicalVolumeId	integer,	// FOREIGN KEY FOR SNRVOLUME TABLE
ReplicationPolicyId	integer,	// FOREIGN KEY FOR ReplicationPolicy TABLE
RVScratchVolumeId	integer,	// FOREIGN KEY FOR RVScratchVolume table
Flags	integer,	
JobId	LONGLONG,	
SnapVolumeId	integer,	// FOREIGN KEY FOR SNRVOLUME TABLE

In the exemplary replication volume table, id is a unique identification number assigned by the system to the snapshot; PointInTime represents the date and time that the snapshot was created; CreationTime represents the date and time that the snapshot was completed; ModifyTime is the recorded date and time of the snapshot taken prior to the current snapshot; CurrentState is an identifier used to indicate a current status of the snapshot (e.g. pending, completed, unfinished, etc.); PrimaryVolumeId is the identifier for the information store **90** from which the snapshot is being made; PhysicalVolumeId is a hardware identifier for the information store **90**; RVScratchVolumeId is an identifier for a scratch volume, which in some embodiments may be used to buffer additional memory as known to those of skill in the art; Flags contains a 32 bit word for various settings such as whether a snapshot has been taken previously, etc.; JobId stores the identifier for the job as assigned by a storage management module; and the SnapVolumeId points to the physical destination storage device **115** to which the snapshot is written.

As each snapshot indexes an information store at a given point in time, a mechanism must be provided that allows the snapshots taken of an information store to be chronologically related so that they are properly used for restoring an information store **90**. According to the replication volume table **102**, the CurrentRole integer may store a value for the relative

8

position of a given snapshot in hierarchy of snapshots taken from a given information store **90** (e.g. first (t_0), second (t_1), t_2 , t_3 , etc.)

In some embodiments, components of the system may reside on and be executed by a single computer. According to this embodiment, a data agent **95**, media agent **105** and storage manager **100** are located at the client computer **85** to coordinate and direct local copying, archiving, migration, and retrieval application functions among one or more storage devices **115** that are remote or distinct from the information store **90**. This embodiment is further described in U.S. patent application Ser. No. 09/610,738.

One embodiment of a method for using the system of the present invention to perform snapshots is illustrated in the flow diagram of FIG. **3**. When the system is initialized, or at other times as directed by a user or rules, e.g., policies or other instructions, the storage manager directs the data agent to perform an initial full snapshot of the data stored in the information store, e.g., indexing the location of all data in the information store, in conjunction with one or more media agents. The system copies all of the data on the information store with the initial snapshot to a storage device, step **300**.

Advantageously, the snapshot and data copied from the information store may be written to a storage device that is remote or different from the information store, step **302**, e.g., local data from a given information store is written to a storage device attached to a network. The selection of a destination storage device for the snapshot may be accomplished using one or more techniques known to those of skill in the art. For example, a fixed mapping may be provided indicating a storage device for which all snapshots and copied or changed data should be written. Alternatively, an algorithm may be implemented to dynamically select a storage device from among a number of storage devices available on a network. For example, a storage manager may select a media agent to handle the transfer of the snapshot and copied data to a specific storage device based on criteria such as available bandwidth, other scheduled storage operations, media availability, storage policies, storage preferences, or other consider considerations. The snapshot, preferably along with the data from the information store, is written to the selected destination storage device, step **304**. According to certain embodiments, the snapshot contains information regarding the files and folders that are tracked by the snapshot. Alternatively, the information regarding the files and folders that are indexed by the snapshot, e.g., file system information, are stored on the storage device.

One embodiment of a snapshot used to track clusters read from the information store to clusters in a snapshot, as well as to map file and folder names corresponding to the snapshot clusters, is illustrated in FIG. **4**. It should be noted that clusters are but one level of granularity that may be indexed by a snapshot, e.g., blocks, extents, etc. During the scan, the data agent creates a snapshot **350** and writes data, e.g., new or changed data, to a storage device **115**. According to the present embodiment, the snapshot is illustrated as a flat file data structure, although those of skill in the art will recognize that the snapshot may be embodied in a number of disparate types of data structures.

The snapshot **350** is used to associate the original cluster numbers from an information store with clusters on a storage device, which in the present embodiment is a magnetic tape. It should be appreciated by those of skill in the art that the present invention is not limited to magnetic tape, and that the systems and methods described herein may be applicable to using snapshots with other storage technologies, e.g., storing

US 7,840,533 B2

9

disk geometry data to identify the location of a cluster on a storage device, such as a hard disk drive.

The tape offsets **356** for the clusters **372** in the snapshot **370** are mapped to original disk cluster information **352**. File and folder names **354** may be scanned from the information store's FAT and also mapped to the tape offsets **356**. A file part column **358** in the snapshot tracks the clusters **372** for each file and folder where each file and folder contains an entry for the first cluster **372**. For files or folders that are stored in more than one cluster, sometimes not in contiguous clusters, the offset table entry for each further cluster is numbered consecutively **358**.

In order to identify the files and folders represented by the stored clusters **372**, e.g., changed data, in the snapshot **370**, the map may exclude data from columns relating to the original disc clusters **352** and last snapshot **360**. In order to keep track of changed verses unchanged clusters, however, the original disk cluster information **352** is stored in the map **350**. Other information may also be stored in the map **350**, such as timestamps for last edit and creation dates of the files.

For each snapshot, even though only clusters that have been changed or created since a previous snapshot are tracked in a given snapshot after the initial snapshot t_0 , the snapshot may be provided with the data from all previous snapshots to provide the latest snapshot with folder and file information such that an index of the entire information store is maintained concurrently each snapshot. Alternatively, this may be bypassed in favor of creating a snapshot that indexes all data at a given point in time in the information store and copying only changed data.

Entries from each snapshot **350** may also contain a last-snapshot field **360** that holds an identifier for the last snapshot containing the cluster indexed by the entry at the time the current snapshot was created. According to an alternative embodiment, e.g., for snapshots that do not store the information from the information store's FAT, the snapshot only tracks clusters stored in the information store with the clusters indexed by the snapshot. For those embodiments, the snapshot **350** contains neither file and folder information **345** nor file part information **358**.

Returning to FIG. 3, once the first full snapshot t_0 has been taken, step **300**, the storage manager may implement a rule, policy, or similar set of instructions that require snapshots to be taken at certain time intervals. Accordingly, at each time interval where a subsequent snapshot t_n is taken, the data agent works in conjunction with one or more of the media agents to perform and store snapshot and accompanying data that changed since the subsequent snapshot, t_{n-1} , loop **306**.

For each snapshot, t_n , that is taken of the information store, a comparison is performed such that only the clusters which have changed or been created since the last snapshot, t_{n-1} , was taken of that volume are stored, step **310**. For example, in some embodiments the data agent employs a block filter or similar construct known to those of skill in the art to compare snapshot t_n with t_{n-1} and thereby detect changed clusters on an information store. Alternatively, the data agent may use other techniques known in the art, such as Copy on Write ("COW"), to identify changed data on an information store. If a given cluster in the information store has changed since the last snapshot in which the cluster appears, or if the cluster from the information store was created subsequent to the last snapshot, then the cluster is read from information store and stored with the new snapshot being written to the storage device, step **314**.

A determination is made regarding the given storage device to which the snapshot and changed data (which may also include newly created data) is to be written, step **316**. Tech-

10

niques such as those described in conjunction with storage of the initial snapshot, steps **302** and **304**, may also be employed regarding storage of subsequent snapshots. Advantageously, the initial snapshot and any subsequent snapshot may be written to any storage device available in the network. Furthermore, there is no limitation to the combination of devices used to store the snapshots for a given information store. For example, an initial snapshot may be written to storage device A, a second and third snapshots may be written to storage device B, and a fourth snapshot may be written to storage device C. Regardless of the storage device that is selected, step **316**, the replication volume table is updated to reflect the location, step **318**, allowing snapshots to be located when a user requests to restore the information store from which the snapshots were taken.

System administrators use stored snapshots, in conjunction with the changed data that the snapshot indexes or tracks, to recover lost or corrupted information. FIG. 5 presents a flow diagram illustrating one embodiment of a method for restoring an information store from one or more snapshots. If the user or a system process wants to restore an information store from one or more snapshots, an interface is presented to restore the snapshot, step **400**. The interface may be, for example, a graphical user interface ("GUI"), and Application Programming Interface ("API") or other interface known to those of skill in the art. The storage manager scans the replication volume table to identify available snapshots for presentation in a menu that allows selection of an available snapshot, step **402**.

When the user selects a snapshot, the storage manager performs a query of the replication volume table to identify all previous snapshots for an information store from which the selected snapshot was taken, step **404**. This may be accomplished by performing a search on the replication volume table for all snapshots with the same PrimaryVolumeld or PhysicalVolumeld. Starting with the selected snapshot, for each snapshot in the query result, loop **406**, the storage manager directs a given media agent, in conjunction with a given data agent, to read and restore all clusters of changed data not already restored from clusters indexed by a prior snapshot, e.g., the latest version of each cluster, step **408**. According to one embodiment, this is accomplished by restoring the clusters indexed by each of the snapshots in the query result, starting with the original snapshot, and overwriting clusters indexed by the original snapshot with changed clusters indexed by subsequent snapshots up to the snapshot representing the point in time selected by the user or system process. As an alternative, the last snapshot field of the selected snapshot may be utilized to determine the snapshots that should be utilized in the restore operation. The latest version of each cluster, starting with those indexed by the selected snapshot, is then restored, step **408**.

As discussed above, embodiments of the invention are contemplated wherein FAT information of the information store is stored in conjunction with a given snapshot, e.g. the file and folder information corresponding to the clusters of changed data indexed by a given snapshot. Accordingly, the storage manager may allow the user to select individual files and/or folders to be selected for restoration from a snapshot. With reference to FIG. 6, a flow diagram is presented illustrating one embodiment of a method for restoring individual files and/or folders indexed by a snapshot.

When the user desires to restore the information store to a given point in time, the user interface allows the user to view the files and folders indexed by a snapshot representing the point in time as if the user were viewing a folder structure on a storage device, step **500**. The storage manager retrieves the

US 7,840,533 B2

11

file and folder information for changed data that is indexed by one or more snapshots for display. Once one or more files and/or folders are selected, step 502, the storage manager selects those snapshots that index the given version of the files and/or folders using the replication volume table, step 502. Each snapshot indexing data for the one or more files to be restored are opened serially, loop 506. The changed data for the selected files and folders that are indexed by the snapshots are restored from clusters indexed by each snapshot, step 508, but not overwriting clusters indexed by prior snapshots.

While the invention has been described and illustrated in connection with preferred embodiments, many variations and modifications as will be evident to those skilled in this art may be made without departing from the spirit and scope of the invention, and the invention is thus not to be limited to the precise details of methodology or construction set forth above as such variations and modification are intended to be included within the scope of the invention.

What is claimed is:

1. A computer-implemented method of creating snapshots for an information store, wherein the information store is coupled via a computer network to at least a first storage medium and a second storage medium, the method comprising:

- performing a first snapshot of data in an information store at a first time;
- selecting the first storage medium for storage of the data associated with the first snapshot;
- copying the data associated with the first snapshot to the first storage medium, wherein the first storage medium is different from the information store;
- tracking changes to the data associated with the first snapshot, the changes occurring between the first time and a second time occurring after the first time;
- performing at least a second snapshot of the data in the information store at the second time;
- selecting the second storage medium for storage of the tracked changes to the data associated with the first snapshot, wherein the second storage medium is different from the information store; and
- copying to the second storage medium the tracked changes to the data associated with the first snapshot, wherein the copying comprises using the second snapshot to identify a location in the information store of the changes to the data associated with the first snapshot.

2. The method of claim 1, wherein tracking changes to the data associated with the first snapshot comprises tracking changes to the data associated with the first snapshot using a block filter to identify changes to the data associated with the first snapshot and then storing data identifying a state of the data associated with the first snapshot in a data structure.

3. The method of claim 1, wherein tracking changes to the data associated with the first snapshot comprises tracking changes to the data associated with the first snapshot using a block filter to identify changes to the data associated with the first snapshot.

4. The method of claim 1, wherein the first storage medium and the second storage medium comprise the same physical storage medium.

5. The method of claim 1, wherein the steps of selecting the first storage medium or selecting the second storage medium comprises:

- communicating with a storage management component to identify a media management component among multiple media management components according to a first selection criteria, wherein the identified media manage-

12

- ment component directs storage operations associated with multiple storage media; and

- communicating with the identified media management component to select a storage medium among the multiple storage media according to a second selection criteria,

- wherein the first selection criteria is related to available bandwidth, other storage operations, storage media availability, storage policies, or storage preferences.

6. The method of claim 1, wherein the steps of selecting the first storage medium or selecting the second storage medium comprises:

- communicating with a storage management component to identify a media management component among multiple media management components according to a first selection criteria, wherein the identified media management component directs storage operations associated with multiple storage media; and

- communicating with the identified media management component to select a storage medium among the multiple storage media according to a second selection criteria.

7. A system for creating snapshots, wherein the system is coupled via a computer network to at least a first storage medium and a second storage medium, the system comprising:

- at least one processor;
- an information store coupled to the processor;
- means for performing a first snapshot of data in the information store at a first time;
- means for selecting the first storage medium for storage of the data associated with the first snapshot;
- means for copying the data associated with the first snapshot to the first storage medium, wherein the first storage medium is different from the information store;
- means for tracking changes to the data associated with the first snapshot, the changes occurring between the first time and a second time occurring after the first time;
- means for performing at least a second snapshot of the data in the information store at the second time;
- means for selecting the second storage medium for storage of the tracked changes to the data associated with the first snapshot, wherein the second storage medium is different from the information store; and
- means for copying to the second storage medium the tracked changes to the data associated with the first snapshot, wherein the copying comprises using the second snapshot to identify a location in the information store of the changes to the data associated with the first snapshot.

8. The system of claim 7, further comprising:

- means for communicating with a storage management component to identify a media management component among multiple media management components according to a first selection criteria, wherein the identified media management component directs storage operations associated with multiple storage media; and
- means for communicating with the identified media management component to select a storage medium among the multiple storage media according to a second selection criteria.

9. A computer-readable storage medium carrying instructions, which when performed by a processor, perform a method of creating snapshots for an information store, wherein the information store is coupled via a computer network to at least a first storage medium and a second storage medium, the method comprising:

US 7,840,533 B2

13

at a first time, performing a first snapshot of data in an information store;
selecting the first storage medium for storage of the data associated with the first snapshot;
copying the data associated with the first snapshot to the first storage medium, wherein the first storage medium is different from the information store;
tracking changes to the data associated with the first snapshot, the changes occurring between the first time and a second time occurring after the first time;
at the second time, performing at least a second snapshot of the data in the information store;
selecting the second storage medium for storage of the tracked changes to the data associated with the first snapshot, wherein the second storage medium is different from the information store; and
copying to the second storage medium the tracked changes to the data associated with the first snapshot, wherein the copying comprises using the second snapshot to identify a location in the information store of the changes to the data associated with the first snapshot.

10. The computer-readable storage medium of claim 9, wherein tracking changes to the data associated with the first snapshot comprises tracking changes to the data associated with the first snapshot using a block filter to identify changes to the data associated with the first snapshot and then storing data identifying a state of the data associated with the first snapshot in a data structure.

11. The computer-readable storage medium of claim 9, wherein tracking changes to the data associated with the first snapshot comprises tracking changes to the data associated with the first snapshot using a block filter to identify changes to the data associated with the first snapshot.

14

12. The computer-readable storage medium of claim 9, wherein the first storage medium and the second storage medium comprise the same physical storage medium.

13. The computer-readable storage medium of claim 9, wherein the steps of selecting the first storage medium or selecting the second storage medium comprises:

communicating with a storage management component to identify a media management component among multiple media management components according to a first selection criteria, wherein the identified media management component directs storage operations associated with multiple storage media; and
communicating with the identified media management component to select a storage medium among the multiple storage media according to a second selection criteria,

wherein the first selection criteria is related to available bandwidth, other storage operations, storage media availability, storage policies, or storage preferences.

14. The computer-readable storage medium of claim 9, wherein the steps of selecting the first storage medium or selecting the second storage medium comprises:

communicating with a storage management component to identify a media management component among multiple media management components according to a first selection criteria, wherein the identified media management component directs storage operations associated with multiple storage media; and

communicating with the identified media management component to select a storage medium among the multiple storage media according to a second selection criteria.

* * * * *

EXHIBIT C

(12) **United States Patent**
Prahlad et al.

(10) **Patent No.:** **US 8,447,728 B2**
(45) **Date of Patent:** ***May 21, 2013**

(54) **SYSTEM AND METHOD FOR STORAGE OPERATION ACCESS SECURITY**

(75) Inventors: **Anand Prahlad**, East Brunswick, NJ (US); **Srinivas Kavuri**, Hyderabad (IN)

(73) Assignee: **CommVault Systems, Inc.**, Oceanport, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/250,997**

(22) Filed: **Sep. 30, 2011**

(65) **Prior Publication Data**
US 2012/0023140 A1 Jan. 26, 2012

Related U.S. Application Data

(63) Continuation of application No. 12/058,511, filed on Mar. 28, 2008, now Pat. No. 8,108,427, which is a continuation of application No. 11/694,784, filed on Mar. 30, 2007.

(60) Provisional application No. 60/852,584, filed on Oct. 17, 2006.

(51) **Int. Cl.**
G06F 7/00 (2006.01)

(52) **U.S. Cl.**
USPC **707/627; 707/785; 726/27**

(58) **Field of Classification Search**
USPC **707/627, 785; 726/27**
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,296,465	A	10/1981	Lemak
4,686,620	A	8/1987	Ng
4,995,035	A	2/1991	Cole et al.
5,005,122	A	4/1991	Griffin et al.
5,093,912	A	3/1992	Dong et al.
5,133,065	A	7/1992	Cheffetz et al.
5,193,154	A	3/1993	Kitajima et al.
5,212,772	A	5/1993	Masters
5,226,157	A	7/1993	Nakano et al.
5,239,647	A	8/1993	Anglin et al.

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0259912	A1	3/1988
EP	0405926	A2	1/1991

(Continued)

OTHER PUBLICATIONS

“Associate”, Collins English Dictionary, London: Collins, 2000, Credo Reference [online][retrieved on Jul. 30, 2009], available at <<http://wNw.credoreference.com/entry/hcengdicVassociate>>, 1 page.

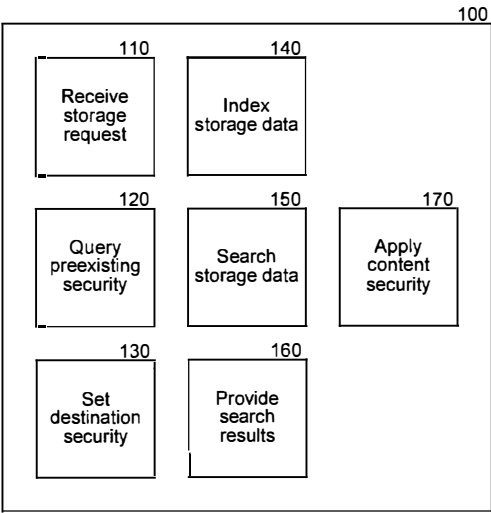
(Continued)

Primary Examiner — Amy Ng
(74) *Attorney, Agent, or Firm* — Perkins Coie LLP

(57) **ABSTRACT**

A method and system for controlling access to stored data is provided. The storage access control system leverages a pre-existing security infrastructure of a system to inform the proper access control that should be applied to data stored outside of its original location, such as a data backup. The storage access control system may place similar access control restrictions on the backup files that existed on the original files. In this way, the backed up data is given similar protection as that of the original data.

22 Claims, 5 Drawing Sheets



US 8,447,728 B2

Page 2

U.S. PATENT DOCUMENTS

5,241,668	A	8/1993	Eastridge et al.	6,356,801	B1	3/2002	Goodman et al.
5,241,670	A	8/1993	Eastridge et al.	6,381,331	B1	4/2002	Kato
5,276,860	A	1/1994	Fortier et al.	6,389,432	B1	5/2002	Pothapragada et al.
5,276,867	A	1/1994	Kenley et al.	6,418,478	B1	7/2002	Ignatius et al.
5,287,500	A	2/1994	Stoppani, Jr.	6,421,711	B1	7/2002	Blumenau et al.
5,301,351	A	4/1994	Jippo et al.	6,487,561	B1	11/2002	●fek et al.
5,311,509	A	5/1994	Heddes et al.	6,519,679	B2	2/2003	Devireddy et al.
5,321,816	A	6/1994	Rogan et al.	6,538,669	B1	3/2003	Lagueux, Jr. et al.
5,333,315	A	7/1994	Saether et al.	6,542,972	B2	4/2003	Ignatius et al.
5,347,653	A	9/1994	Flynn et al.	6,564,228	B1	5/2003	●"Connor
5,410,700	A	4/1995	Fecteau et al.	6,577,734	B1	6/2003	Etzel et al.
5,448,724	A	9/1995	Hayashi	6,604,149	B1	8/2003	Deo et al.
5,491,810	A	2/1996	Allen	6,654,825	B2	11/2003	Clapp et al.
5,495,607	A	2/1996	Pisello et al.	6,658,526	B2	12/2003	Nguyen et al.
5,504,873	A	4/1996	Martin et al.	6,772,332	B1	8/2004	Boebert et al.
5,544,345	A	8/1996	Carpenter et al.	6,898,286	B2	5/2005	Murray
5,544,347	A	8/1996	Yanai et al.	6,973,621	B2	12/2005	Sie et al.
5,559,957	A	9/1996	Balk	7,035,880	B1	4/2006	Crescenti et al.
5,559,991	A	9/1996	Kanfi	7,130,970	B2	10/2006	Devassy et al.
5,598,546	A	1/1997	Blomgren	7,209,972	B1	4/2007	Ignatius et al.
5,619,644	A	4/1997	Crockett et al.	7,213,269	B2	5/2007	●rthlieb et al.
5,638,509	A	6/1997	Dunphy et al.	7,277,941	B2	10/2007	Ignatius et al.
5,673,381	A	9/1997	Huai et al.	7,287,045	B2	10/2007	Saika et al.
5,683,513	A	11/1997	Fujimaki	7,287,047	B2	10/2007	Kavuri
5,699,361	A	12/1997	Ding et al.	7,315,923	B2	1/2008	Retnanuma et al.
5,729,743	A	3/1998	Squibb	7,320,068	B2	1/2008	Zimmiewicz et al.
5,751,997	A	5/1998	Kullick et al.	7,328,189	B2	2/2008	Ling
5,752,041	A	5/1998	Fosdick	7,360,252	B1	4/2008	Torrubia-Saez et al.
5,758,068	A	5/1998	Brandt et al.	7,389,273	B2	6/2008	Irwin et al.
5,758,359	A	5/1998	Saxon	7,395,282	B1	7/2008	Crescenti et al.
5,761,677	A	6/1998	Senator et al.	7,401,154	B2	7/2008	Ignatius et al.
5,761,734	A	6/1998	Pfeffer et al.	7,506,102	B2	3/2009	Lev-Ran et al.
5,764,972	A	6/1998	Crouse et al.	7,519,827	B2	4/2009	Anderson et al.
5,778,395	A	7/1998	Whiting et al.	7,581,077	B2	8/2009	Ignatius et al.
5,805,920	A	9/1998	Sprenkle et al.	7,620,976	B2	11/2009	Low et al.
5,812,398	A	9/1998	Nielsen	7,627,569	B2	12/2009	Gafer
5,813,009	A	9/1998	Johnson et al.	7,627,776	B2	12/2009	Petruzzo
5,813,017	A	9/1998	Morris	7,702,693	B1	4/2010	Aiyagari et al.
5,860,104	A	1/1999	Witt et al.	7,739,381	B2	6/2010	Ignatius et al.
5,875,478	A	2/1999	Blumenau	7,748,027	B2	6/2010	Patrick
5,887,134	A	3/1999	Ebrahim	7,761,713	B2	7/2010	Baar
5,901,327	A	5/1999	●fek	7,782,742	B2	8/2010	Park
5,924,102	A	7/1999	Perks	7,805,600	B2	9/2010	Bucher et al.
5,950,205	A	9/1999	Aviani, Jr.	7,818,262	B2	10/2010	Kavuri et al.
5,956,519	A	9/1999	Wise et al.	7,840,537	B2	11/2010	Gokhale et al.
5,970,233	A	10/1999	Liu et al.	7,882,315	B2	2/2011	Tsai et al.
5,970,255	A	10/1999	Tran et al.	7,926,087	B1	4/2011	Holl, II et al.
5,974,563	A	10/1999	Beeler, Jr.	8,108,427	B2	1/2012	Prahlad et al.
5,999,629	A	12/1999	Heer et al.	8,131,648	B2	3/2012	Barton
6,003,089	A	12/1999	Shaffer et al.	8,165,221	B2	4/2012	Zheng et al.
6,009,274	A	12/1999	Fletcher et al.	8,200,191	B1	6/2012	Belser et al.
6,012,090	A	1/2000	Chung et al.	2002/0007347	A1	1/2002	Blumenthal et al.
6,021,415	A	2/2000	Cannon et al.	2002/0007351	A1	1/2002	Hilleagass et al.
6,026,414	A	2/2000	Anglin	2002/0077988	A1	6/2002	Sasaki et al.
6,052,735	A	4/2000	Ulrich et al.	2002/0120726	A1	8/2002	Padole et al.
6,076,148	A	6/2000	Kedem	2002/0128976	A1	9/2002	●"Connor et al.
6,094,416	A	7/2000	Ying	2002/0147734	A1	10/2002	Shoup et al.
6,094,684	A	7/2000	Pallmann	2002/0174011	A1	11/2002	Sanchez et al.
6,105,129	A	8/2000	Meier et al.	2003/0005428	A1	1/2003	Roman
6,131,095	A	10/2000	Low et al.	2003/0200104	A1	10/2003	Heming et al.
6,131,190	A	10/2000	Sidwell	2004/0093229	A1	5/2004	Plain
6,148,412	A	11/2000	Cannon et al.	2004/0210509	A1	10/2004	Eder
6,154,787	A	11/2000	Urevig et al.	2004/0249759	A1	12/2004	Higashi et al.
6,161,111	A	12/2000	Mutalik et al.	2004/0255143	A1	12/2004	Wemyss et al.
6,167,402	A	12/2000	Yeager	2005/0027657	A1	2/2005	Leontiev et al.
6,169,976	B1	1/2001	Colosso	2005/0091655	A1	4/2005	Probert et al.
6,212,512	B1	4/2001	Barney et al.	2005/0097440	A1	5/2005	Lusk et al.
6,260,069	B1	7/2001	Anglin	2005/0108526	A1	5/2005	Robertson
6,269,431	B1	7/2001	Dunham	2006/0224846	A1	10/2006	Amarendran et al.
6,275,953	B1	8/2001	Vahalia et al.	2006/0242296	A1	10/2006	Woolard et al.
6,292,783	B1	9/2001	Rohler et al.	2007/0198421	A1	8/2007	Muller et al.
6,301,592	B1	10/2001	Aoyama et al.	2007/0198422	A1	8/2007	Prahlad et al.
6,324,581	B1	11/2001	Xu et al.	2008/0005380	A1	1/2008	Kawasaki et al.
6,328,766	B1	12/2001	Long	2008/0091747	A1	4/2008	Prahlad et al.
6,330,570	B1	12/2001	Crighton	2008/0229037	A1	9/2008	Bunte et al.
6,330,642	B1	12/2001	Carteau	2008/0243795	A1	10/2008	Prahlad et al.
6,343,324	B1	1/2002	Hubis et al.	2008/0307020	A1	12/2008	Ko et al.
RE37,601	E	3/2002	Eastridge et al.	2008/0320319	A1	12/2008	Muller et al.
				2009/0222907	A1	9/2009	Guichard

US 8,447,728 B2

Page 3

2009/0319534 A1 12/2009 Gokhale
 2009/0319585 A1 12/2009 Gokhale
 2010/0031017 A1 2/2010 Gokhale et al.
 2010/0242096 A1 9/2010 Varadharajan et al.
 2010/0313039 A1 12/2010 Ignatius et al.

FOREIGN PATENT DOCUMENTS

EP 0467546 A2 1/1992
 EP 0774715 A1 5/1997
 EP 0809184 A1 11/1997
 EP 0862304 A2 9/1998
 EP 0899662 A1 3/1999
 EP 0981090 A1 2/2000
 WO WO-9513580 A1 5/1995
 WO WO-9912098 3/1999

OTHER PUBLICATIONS

"Right", Chambers 21st Century Dictionary, London: Chambers Harrap, 2001, Credo Reference [online] [retrieved on Jul. 30, 2009], available at <<http://wNw.credoreference.com/entry/chambdict/right>>, 1 page.

Armstead et al., "Implementation of a Campus-wide Distributed Mass Storage Service: The Dream vs. Reality," *IEEE*, 1995, pp. 190-199.

Arneson, "Mass Storage Archiving in Network Environments," Digest of Papers, Ninth IEEE Symposium on Mass Storage Systems, Oct. 31, 1988-Nov. 3, 1988, pp. 45-50, Monterey, CA.

Cabrera et al., "ADSM: A Multi-Platform, Scalable, Backup and Archive Mass Storage System," Digest of Papers, Compcon '95, Proceedings of the 40th IEEE Computer Society International Conference, Mar. 5, 1995-Mar. 9, 1995, pp. 420-427, San Francisco, CA. CommVault Systems, Inc., "Netix Books Online Documentation," submitted on CD-ROM, released Dec. 2005.

CommVault, "Firewall Considerations—How to," <http://documentation.commvault.com/commvault/release_8.0.0/books_online_1/english_us/features/firewall/firewall_how_to.htm>, internet accessed on Feb. 27, 2009, 11 pages.

CommVault, "Firewall Considerations," <http://documentation.commvault.com/commvault/release_8.0.0/books_online_1/english_us/features/firewall/firewall.htm>, internet accessed on Feb. 27, 2009, 8 pages.

Eitel, "Backup and Storage Management in Distributed Heterogeneous Environments," *IEEE*, 1994, pp. 124-126.

Jander, M., "Launching Storage-Area Net," *Data Communications*, US, McGraw Hill, NY, vol. 27, No. 4 (Mar. 21, 1998), pp. 64-72.

Jason Gait, "The Optical File Cabinet: A Random-Access File System for Write-Once Optical Disks," *IEEE Computer*, vol. 21, No. 6, pp. 11-22 (1988) (see in particular figure 5 in p. 15 and recitation in claim 5).

Kwok, S. H., Digital rights management for the online music business, SIGecom Exch. 3, 3 (Jun. 2002), available at <<http://doi.acm.org/10.1145/844339.844347>>, 8 pages.

Microsoft SQL Server Documentation, "Adding a Member to a Pre-defined Role," 1988-2000, accessed Apr. 18, 2008, 3 pages.

Microsoft SQL Server Documentation, "Adding a Member to a SQL Server Database Role," 1988-2000, accessed Apr. 18, 2008, 2 pages.

Microsoft SQL Server Documentation, "Adding a Windows User or Group," 1988-2000, accessed Apr. 18, 2008, 2 pages.

Microsoft SQL Server Documentation, "Authentication Modes," 1988-2000, accessed Apr. 18, 2008, 4 pages.

Microsoft SQL Server Documentation, "Creating User-Defined SQL Server Database Roles," 1988-2000, accessed Apr. 18, 2008, 2 pages.

Microsoft SQL Server Documentation, "Granting a Windows User or Group Access to a Database," 1988-2000, accessed Apr. 18, 2008, 2 pages.

Microsoft SQL Server Documentation, "Security Architecture," 1988-2000, accessed Apr. 18, 2008, 1 page.

Ron White, "How Computers Work", Sixth Edition, Que Corporation, Jun. 26, 2002, 7 pages.

Rosenblum et al., "The Design and Implementation of a Log-Structured File System," *Operating Systems Review SIGOPS*, vol. 25, No. 5, New York, US, pp. 1-15 (May 1991).

U.S. Patent

May 21, 2013

Sheet 1 of 5

US 8,447,728 B2

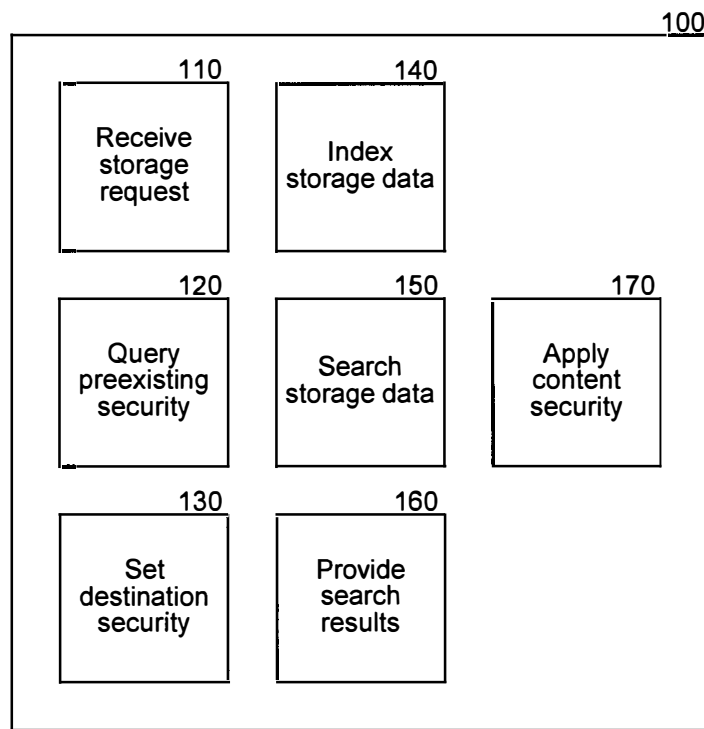


FIG. 1

U.S. Patent

May 21, 2013

Sheet 2 of 5

US 8,447,728 B2

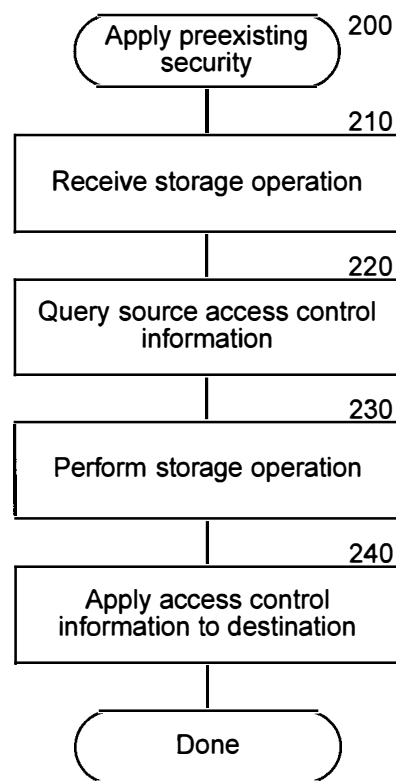


FIG. 2

U.S. Patent

May 21, 2013

Sheet 3 of 5

US 8,447,728 B2

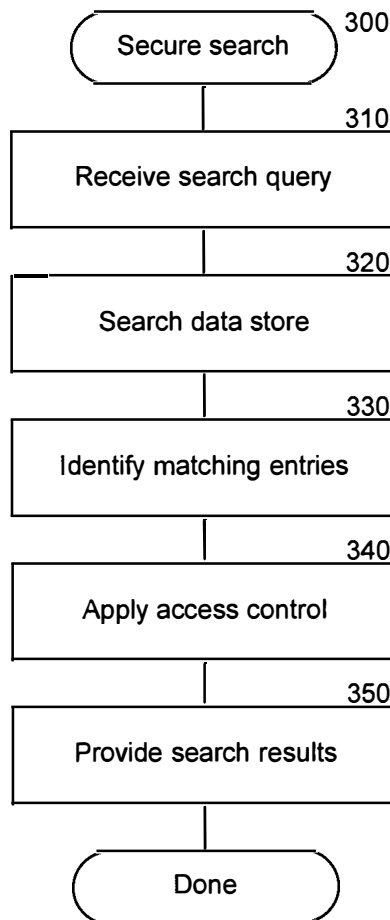


FIG. 3

U.S. Patent

May 21, 2013

Sheet 4 of 5

US 8,447,728 B2

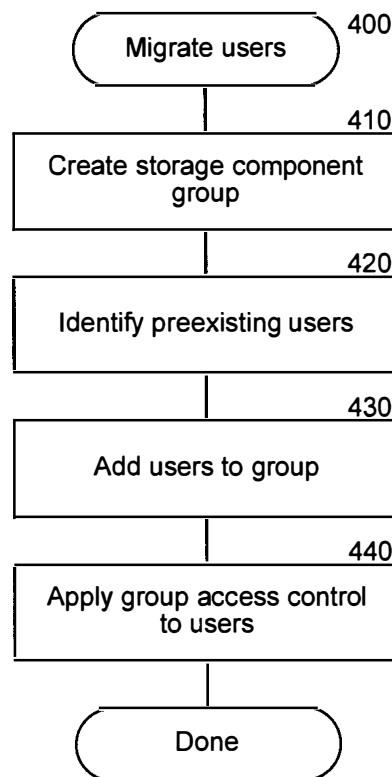


FIG. 4

U.S. Patent

May 21, 2013

Sheet 5 of 5

US 8,447,728 B2

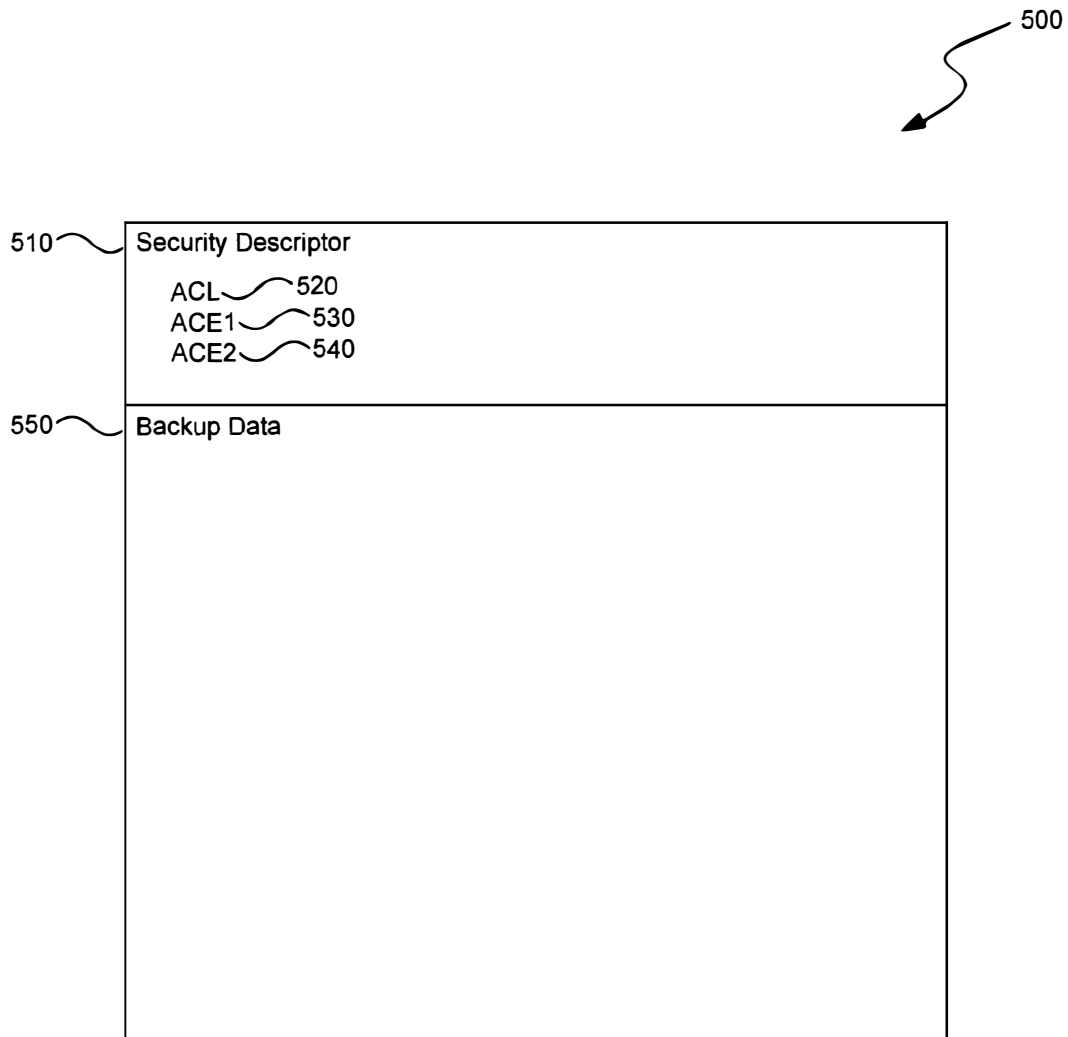


FIG. 5

US 8,447,728 B2

1

**SYSTEM AND METHOD FOR STORAGE
OPERATION ACCESS SECURITY****CROSS-REFERENCE TO RELATED
APPLICATIONS**

The present application is a continuation of U.S. application Ser. No. 12/058,511 entitled "SYSTEM AND METHOD FOR STORAGE OPERATION ACCESS SECURITY" and filed on Mar. 28, 2008, now U.S. Pat. No. 8,108,427 which is a continuation of U.S. application Ser. No. 11/694,784 entitled "SYSTEM AND METHOD FOR STORAGE OPERATION ACCESS SECURITY" and filed on Mar. 30, 2007, which claims priority to U.S. Provisional Application No. 60/852,584 entitled "METHOD AND SYSTEM FOR COLLABORATIVE SEARCHING," and filed on Oct. 17, 2006, each of which is hereby incorporated by reference.

BACKGROUND

Traditional security systems operate on the principal of limiting access to data. Each user of the system is generally identified with a user name, and access rights are assigned to each user. For example, users may be permitted or prevented from accessing certain files or adding new hardware to a computer system. Users may also be assigned to groups where each member of the group is given common access rights. Often a great amount of administrative effort has been put into creating users and groups and assigning them appropriate access rights in a traditional computer security system. For example, Microsoft Windows provides Active Directory for creating users and groups and assigning access to resources throughout a computer network. File systems also often provide access control. For example, the NT File System (NTFS) provides folder and file access based on user and group identifiers and the type of access requested such as read, write, execute, and other operations. An organization may have an extensive scheme of groups and access rights. For example, there may be a group of accounting department users that have different rights than engineering department users. The organization may also have identified certain users as administrators that have additional rights to administer the system.

Computer systems contain large amounts of personal data, such as financial data, names, addresses, telephone numbers, bank account information, photographs and much more. Corporate computer systems often contain confidential information, such as trade secrets, manufacturing processes, business strategy, and so on. With the increased reliance on computer systems to store critical information, the importance of protecting this data against loss has grown. For example, traditional storage management systems receive an identification of a file location of an original file and then create one or more secondary copies, such as backup files, containing the contents of the original file. These secondary copies can then later be used to restore the original data should anything happen to the original data. Secondary copies of data are often stored in a publicly accessible location for quick restoration of data in the event of a disaster or other data loss event. For example, backup files may be stored on a widely accessible server, and tapes and other media used for storing backup files may be physically accessible to many users.

Backed up data may contain sensitive information that is more widely accessible than the original data. Backing up data often removes the data from the well-planned security environment in which it was originally stored. Even though a system administrator may have gone to great lengths to prop-

2

erly limit access to data throughout a network, once the data is stored as one or more secondary copies it is often more accessible than originally intended. For example, the CEO of a company may have many sensitive files on a computer system that only he can access, but if that computer system is backed up, then the backup files may allow unauthorized users to have access to data that they would not normally be able to access. In addition, some systems provide searches based on backup data in which the backup data is indexed. Indexed content does not have the protections imposed on the original files.

There is a need for a system that overcomes the above problems, as well as providing additional benefits.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram that illustrates components of a storage access control system in one embodiment.

FIG. 2 is a flow diagram that illustrates processing of the system to apply preexisting security to data objects in one embodiment.

FIG. 3 is a flow diagram that illustrates processing of the system to perform a secure search in one embodiment.

FIG. 4 is a flow diagram that illustrates processing of the system to migrate users from a preexisting security infrastructure to a storage component security infrastructure in one embodiment.

FIG. 5 is a data structure diagram that illustrates storing of access control information with storage data in one embodiment.

In the drawings, the same reference numbers and acronyms identify elements or acts with the same or similar functionality for ease of understanding and convenience. To easily identify the discussion of any particular element or act, the most significant digit or digits in a reference number refer to the Figure number in which that element is first introduced (e.g., element 1104 is first introduced and discussed with respect to FIG. 11).

The headings provided herein are for convenience only and do not necessarily affect the scope or meaning of the claimed invention.

DETAILED DESCRIPTION**Overview**

A method and system for controlling access to stored data described below leverages a preexisting security infrastructure to inform proper access control that should be applied to data stored outside of its original location, such as a data backup. In one embodiment, the storage access control system receives a request to perform a storage operation that makes data at a source location available at a destination location. For example, the request may indicate that data stored on one computer should be copied stored on a second computer. A storage operation may include many types of operations such as backup, migration, replication, snapshot, hierarchical storage management (HSM), and so on. For example, the storage operation may be a request to make a snapshot copy of data at the source location. The source location may contain electronic information such as file system data objects, application data objects, or other types of storage data objects. Upon receiving the request, the storage access control system queries the source or other location for access control information. For example, if the data includes one or more files, then the storage access control system may examine the file system to determine what access control scheme is currently in place for the data. The file system may

US 8,447,728 B2

3

contain access information that identifies the users and groups that have access to the data. One manner in which the access information may be associated with the data is by storing the access information along with the file. Then, the storage access control system applies the access control information to the data stored at the destination location. For example, the storage access control system may associate the access control information with the data stored at the destination location in a different manner, such as by storing metadata describing the access control information in a content indexing system. In the example of backing up files, the storage access control system may place similar access control restrictions on the backup files that existed on the original files. In this way, the backed up data is given similar protection as that of the original data.

The invention will now be described with respect to various embodiments. The following description provides specific details for a thorough understanding of, and enabling description for, these embodiments of the invention. However, one skilled in the art will understand that the invention may be practiced without these details. In other instances, well-known structures and functions have not been shown or described in detail to avoid unnecessarily obscuring the description of the embodiments of the invention.

The terminology used in the description presented below is intended to be interpreted in its broadest reasonable manner, even though it is being used in conjunction with a detailed description of certain specific embodiments of the invention. Certain terms may even be emphasized below; however, any terminology intended to be interpreted in any restricted manner will be overtly and specifically defined as such in this Detailed Description section.

Improving Security with ACLs and Active Directory

In some embodiments, the storage access control system determines the access control information stored by the pre-existing security infrastructure based on an offline or secondary copy of the data. An offline copy can be a backup, snapshot, or other copy of the data that is not actively being used by a live data server or other computers system. By using a secondary copy, the storage access control system can avoid interrupting user access to the live data by not consuming additional resources on the server or other computer system storing the live copy of the data.

When a live or production copy of the source data is used to create a secondary copy, the preexisting security information associated with the data may also be associated with the secondary copy. For example, if the source data is a file, then the security information associated with the file may be captured when the secondary copy is created and stored with the file or in another location that is associated with the secondary copy. For example, many file systems contain hierarchical security schemes such that access control information applied to a parent file system object (e.g., a folder) is applied to each of the child file system objects (e.g., files in the folder). The storage access control system captures this information so that the access control information applied to source data can also be applied to secondary copies of the source data. For example, if the user later performs a search and the storage access control system searches offline copies of data, then the storage access control system can ensure that the user has similar access (both permitting allowed operations and denying excluded operations) to the offline data that the user had to the original live data from which the offline data was created. For example, if the user could not browse particular source data, then the storage access control system may exclude references to secondary copies of the source data from search results. Similarly, if the user could browse and read a file but

4

not write to it, then the storage access control system may permit the user to receive the file in search results, read from the file, but not make modifications to the file.

In some embodiments, the storage access control system stores access control information as metadata that identifies users or groups authorized to perform storage operations. For example, backup files may contain metadata that lists the users that can access the data contained in the backup file. Alternatively or additionally, backup data that is indexed for searching may be associated with metadata stored with the index to apply access control information in response to search queries. For example, a user that does not have permission to access a particular backup data object may be prevented from receiving that data object in a list of results from a search query, even though the backup data object may satisfy the search criteria. Alternatively or additionally, the user may be able to receive the data object in a list of search results, but not be able to open or view the data object. A data object could be a file system object (e.g., a file or folder), an application data object (e.g., an email mailbox, word processing document, etc.), or other object containing data.

In some embodiments, the storage access control system stores access control information as an Access Control List (ACL) containing Access Control Entries (ACE). The ACL contains a list of users and/or groups that are allowed to access a data object, type of data object, or resource containing a data object. Each ACE may specify a user, group, or other entity that has access to the data object associated with the ACL. In some embodiments, an ACL may contain a list of users or groups that are specifically denied access to a data object. In this way, administrators can apply access control rights in the manner that is most logical for their organization. For example, if everyone in the accounting department except User A should have access to a particular data object, then an administrator may create an ACL associated with the data object containing an ACE that allows access to the accounting department group, and another ACE that denies access to User A. The ACL may also contain Boolean operators that describe combinations of permissions and users that should be applied to a data object.

When a user, system, or process attempts to access a data object, such as to perform a storage operation on the data object, the storage access control system accesses the ACL and associated ACEs related to the data object to determine whether the user has the appropriate access to perform the operation on the data object. If the user has the appropriate access, then permission to perform the operation is granted, and the operation proceeds. If the user does not have the appropriate access, then the storage access control system denies permission to perform the operation, and an error or other information may be conveyed to the user indicating that the operation was not performed.

In some embodiments, the storage access control system further protects secondary copies of data, such as by encrypting the data. This may be useful when the backup data is expected to be stored offsite, such as by a public remote backup provider. The data may be encrypted such that it can only be decrypted by those users or groups with access to the original data. For example, the data may be encrypted using a key that is associated with a particular group of users that has access to the data. Users that are not part of the group will not know or be associated with the key and therefore will not be able to decrypt the data, while users within the group will know the key and can decrypt and access the data.

In some embodiments, the storage access control system assigns access rights based on the content of or metadata associated with a data object, such as by querying a content or

US 8,447,728 B2

5

metadata indexing system. For example, some users may be denied access to files that contain the word “confidential.” An access group of company executives can be granted exclusive access to files that contain the term “board of directors.” The system may apply such content filtering to the data directly, or the system can filter searches for data objects such that the search results do not contain content to which the searching user has not been granted the right to access.

Active Directory Integration for User Creation

In some embodiments, the storage access control system provides a separate security infrastructure, but recognizes users and groups created in the preexisting security infrastructure. For example, server systems running Microsoft Windows often use Active Directory or other systems to create users and groups and assign access rights to those users and groups. The storage access control system may allow creating a separate set of users and groups that are assigned various storage operation rights. However, rather than recreating each user from the Active Directory in the storage access control system, the storage access control system may allow adding an Active Directory user or group to a storage access control system group. For example, when an Active Directory user is added to a storage access control system group, the storage access control system may query the Active Directory to determine information about the user and the access rights associated with the user. Thus, it is not necessary to give storage system operators permissions to create new storage access control system users, and it is not necessary to duplicate the users in both security systems. Similarly, other preexisting security infrastructures could be used with the storage access control system.

The storage access control system may also retrieve other information from the preexisting security system. For example, the preexisting security system may maintain a list of computers associated with a particular user, and the storage access control system can grant the user access, for example, to backup computers in that list. The preexisting security system may contain other supplemental information, such as the user’s email address that the storage access control system may use, for example, to email the user if a storage operation fails. The integration and connection of the storage access control system with the preexisting security system allows the storage access control system to provide a system administrator with additional value in the administrator’s investment of time and resources in the preexisting security system and reduces the need for a redundant investment of time and resources in another security system.

Security-Based Queries and Access Filtering

In some embodiments, the storage access control system provides an indexing and search facility that allows searching based on keywords within backed up documents. The storage access control system stores access control information for indexed files and applies access control to search queries initiated by a user, system, or process. For example, an administrator may be able to search backup data for all users, whereas another user may only be able to search her own backup data. Likewise, an executive of a company may be able to search for and view content containing sensitive business plans or trade secrets, but other employees may not.

Such access control may be applied using ACLs and Active Directory groups as described above. For example, a user with an ACL on an original file that allows the user to view the file can also view search results containing the file, whereas a user without access to the original file cannot view the file by opening it from a list of search results. Similarly, a user that is a member of an Active Directory group that has access to a file will have access to view search results containing the file. In

6

this way, an organization can leverage the investment in an existing security infrastructure to provide similar security for content accessible via a search facility.

Figures

Unless described otherwise below, aspects of the invention may be practiced with conventional systems. Thus, the construction and operation of the various blocks shown in FIG. 1 may be of conventional design, and need not be described in further detail herein to make and use the invention, because such blocks will be understood by those skilled in the relevant art. One skilled in the relevant art can readily make any modifications necessary to the blocks in FIG. 1 (or other embodiments or Figures) based on the detailed description provided herein.

FIG. 1 is a block diagram that illustrates components of the storage access control system in one embodiment. The storage access control system 100 contains a receive storage request component 110, a query preexisting security component 120, a set destination security component 130, an index storage data component 140, a search storage data component 150, a provide search results component 160, and an apply content security component 170. The receive storage request component 110 handles incoming storage requests. For example, a storage request may include a request to copy data from a source location to a destination location. The query preexisting security component 120 queries access control information from an existing security provider external to the storage access control system. For example, files stored in an NTFS file system contain or are otherwise associated with access control information that specifies the users that are allowed to access the file.

The set destination security component 130 applies access control information identified from an external security provider to data managed by the storage access control system. For example, during a backup operation, access control information from a source file is associated with secondary copies that store information from the source file, such that a user has similar access rights to the source file and the backup data. The index storage data component 140 creates an index of storage data managed by the storage access control system. For example, the system 100 can maintain an index of data present in a set of files that have been backed up.

The search storage data component 150 performs searches of indexed storage data to identify matching data objects. The provide search results component 160 prepares identified matching data objects for display to a user. For example, data objects for which the searching user does not have access rights may be removed from the search results before the results are returned to the user. The apply content security component 170 applies security to a data object based on the content of the data object. For example, if a user has not been granted access to documents containing the word “confidential,” then the apply content security component 170 prevents the user from accessing a document containing “confidential.”

FIG. 1 and the following discussion provide a brief, general description of a suitable computing environment in which the invention can be implemented. Although not required, aspects of the invention are described in the general context of computer-executable instructions, such as routines executed by a general-purpose computer, e.g., a server computer, wireless device or personal computer. Those skilled in the relevant art will appreciate that the invention can be practiced with other communications, data processing, or computer system configurations, including: Internet appliances, hand-held devices (including personal digital assistants (PDAs)), wearable computers, all manner of cellular or mobile phones,

US 8,447,728 B2

7

multi-processor systems, microprocessor-based or program-
mable consumer electronics, set-top boxes, network PCs,
mini-computers, mainframe computers, and the like. Indeed,
the terms “computer,” “host,” and “host computer” are gen-
erally used interchangeably herein, and refer to any of the
above devices and systems, as well as any data processor.

Aspects of the invention can be embodied in a special
purpose computer or data processor that is specifically pro-
grammed, configured, or constructed to perform one or more
of the computer-executable instructions explained in detail
herein. Aspects of the invention can also be practiced in
distributed computing environments where tasks or modules
are performed by remote processing devices, which are
linked through a communications network, such as a Local
Area Network (LAN), Wide Area Network (WAN), or the
Internet. In a distributed computing environment, program
modules may be located in both local and remote memory
storage devices.

Aspects of the invention may be stored or distributed on
computer-readable media, including magnetically or opti-
cally readable computer discs, hard-wired or preprogrammed
chips (e.g., EEPROM semiconductor chips), nanotechnology
memory, biological memory, or other data storage media.
Indeed, computer implemented instructions, data structures,
screen displays, and other data under aspects of the invention
may be distributed over the Internet or over other networks
(including wireless networks), on a propagated signal on a
propagation medium (e.g., an electromagnetic wave(s), a
sound wave, etc.) over a period of time, or they may be
provided on any analog or digital network (packet switched,
circuit switched, or other scheme). Those skilled in the rel-
evant art will recognize that portions of the invention reside
on a server computer, while corresponding portions reside on
a client computer such as a mobile or portable device, and
thus, while certain hardware platforms are described herein,
aspects of the invention are equally applicable to nodes on a
network.

FIGS. 2-4 are representative flow diagrams that depict
processes used in some embodiments. These flow diagrams
do not show all functions or exchanges of data, but instead
they provide an understanding of commands and data
exchanged under the system. Those skilled in the relevant art
will recognize that some functions or exchange of commands
and data may be repeated, varied, omitted, or supplemented,
and other (less important) aspects not shown may be readily
implemented.

FIG. 2 is a flow diagram that illustrates the processing of
the system to apply preexisting security to data objects in one
embodiment. These steps are invoked when a storage opera-
tion is performed that results in data being moved or copied
from a source location to a destination location. In step 210,
the system receives a storage operation, such as a request to
copy data from a source location to a destination location. In
step 220, the system queries the source location for access
control information. For example, if the source information is
a file, then the storage access control system queries access
control information from the file system. In step 230, if the
access control information indicates that the requestor of the
storage operation has permission to perform the operation,
then the system performs the requested storage operation. For
example, if the operation is a backup, then the system backs
up data from the source location to the destination location. In
step 240, the system applies the access control information to
the destination data objects, such as backup files or folders.
Access control information captured from a file system may
be stored as metadata in a content indexing system that con-
trols access to secondary copies of the source data. For

8

example, ACLs and ACEs associated with files may be stored
in the content indexing system or otherwise associated with
secondary copies of the files. After step 240, these steps
conclude.

FIG. 3 is a flow diagram that illustrates the processing of
the system to perform a secure search in one embodiment.
These steps are invoked when a user attempts to search for
data objects matching specified criteria. In step 310, the sys-
tem receives a search query specifying the criteria (e.g., of the
data objects) for which the user is searching. For example, the
criteria may contain a file name or the contents of a file that the
user is seeking. In step 320, the system searches one or more
data stores or an index of content of the data stores using the
received query. The system may only search certain data
stores based on the access permitted to the user. The data store
may be a destination location where the data objects were
copied following a storage operation, or the data store may
contain metadata about the data objects, which may be stored
elsewhere. In step 330, the system identifies matching data
object entries in the data store that satisfy the received search
criteria. In step 340, the system applies access control settings
to the search results. For example, certain users may not have
access to documents from a certain location or containing
certain keywords. As another example, the access control
information may be used to decrypt an encrypted search
result. The system may perform the search in two passes.
During the first pass, the system performs a coarse search in
which all data stores to which the user has accessed are
searched to create a list of search results. During the second
pass, a finer grained search of the individual results is per-
formed to determine which search results the user has access
to receive. Search results that the user does not have access to
receive may be removed or replaced with a no access indica-
tor (e.g., an icon) before the search results are displayed to the
user. In step 350, the system provides the search results to the
querying user. After step 350, these steps conclude.

FIG. 4 is a flow diagram that illustrates the processing of
the system to migrate users or security information associated
with users from a preexisting security infrastructure to a
storage application (or component) in one embodiment.
These steps are invoked when, for example, an administrator
manages storage access control for a storage application. In
step 410, the system creates a group within the storage appli-
cation. For example, the administrator may create a group of
users called “Backup Users” that have the necessary access
rights to perform a backup of certain data within the system.
Alternatively, the system may import whole groups from the
preexisting security infrastructure and assign access rights to
the groups and entities within the groups. In step 420, the
system identifies preexisting users that are external to the
storage application. For example, an administrator may have
previously defined the users in the Windows Active Directory
or in another external security component.

In step 430, the system adds the external users to the
storage application group, such that the storage application
group contains users that were not created using the storage
application. For example, a user “Bob Jones” created in the
Active Directory may be added to a group “System Admin-
istrators” within the storage application. The external users
may also be user groups, such that group previously created
by the administrator using Windows Active Directory is
added to the storage application group. In step 440, the system
applies the access control rights of the storage application
group to the added external users. The system is more secure
than traditional systems because each administrator is not
given access to create new users within the storage applica-
tion. By allowing an administrator to add external users to the

US 8,447,728 B2

9

storage application, the system does not need to allow most administrators to have the access rights necessary to create new users within the storage application. For example, an administrator may only be able to add existing users or groups to the storage application. Thus, an administrator of the pre-existing security system can restrict the entities to which an administrator of the storage application can assign rights. Storage system administrators often have access to some of a corporation's most important data, so the ability to control which users can perform storage operations can significantly enhance data security. After step 440, these steps conclude.

FIG. 5 is a data structure diagram that illustrates access control information metadata that may be stored with storage data in one embodiment. The data structure 500 contains a security descriptor 510 and secondary data 550. The security descriptor 510 contains an access control list 520 that specifies the entities that have access to the backup data. The security descriptor 510 may contain multiple access control lists that define different types of access such as read, write, or execute permissions. The access control list 510 contains access control entries ACE1 and ACE2 (shown with respective reference numerals 530 and 540). Each access control entry refers to a different entity, such as a user, group, resource, or other entity, that has some type of access or lack of access to the secondary data 550. Alternatively or additionally, the access control entries 530 and 540 may specify different entries that are denied access to the secondary data 550. The security descriptor 510 may contain other information such as keywords that members of the specified access control list have access rights to. For example, the security descriptor 510 may indicate that a particular entity does not have access to documents containing the keyword "confidential."

Conclusion

From the foregoing, it will be appreciated that specific embodiments of the storage access control system have been described herein for purposes of illustration, but that various modifications may be made without deviating from the spirit and scope of the invention. For example, although certain preexisting security systems have been described, the storage access control system is compatible with any preexisting security system, such as Linux Kerberos, Lightweight Directory Access Protocol (LDAP)-based systems, and others. Although backups have been described, the storage access control system can be applied to other storage operations such as migrating data from one system to another. Accordingly, the invention is not limited except as by the appended claims.

Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense; that is to say, in the sense of "including, but not limited to." The word "coupled", as generally used herein, refers to two or more elements that may be either directly connected, or connected by way of one or more intermediate elements. Additionally, the words "herein," "above," "below," and words of similar import, when used in this application, shall refer to this application as a whole and not to any particular portions of this application. Where the context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number respectively. The word "or" in reference to a list of two or more items, that word covers all of the following interpretations of the word: any of the items in the list, all of the items in the list, and any combination of the items in the list.

The above detailed description of embodiments of the invention is not intended to be exhaustive or to limit the

10

invention to the precise form disclosed above. While specific embodiments of, and examples for, the invention are described above for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. For example, while processes or blocks are presented in a given order, alternative embodiments may perform routines having steps, or employ systems having blocks, in a different order, and some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified. Each of these processes or blocks may be implemented in a variety of different ways. Also, while processes or blocks are at times shown as being performed in series, these processes or blocks may instead be performed in parallel, or may be performed at different times.

The teachings of the invention provided herein can be applied to other systems, not necessarily the system described above. The elements and acts of the various embodiments described above can be combined to provide further embodiments.

These and other changes can be made to the invention in light of the above Detailed Description. While the above description details certain embodiments of the invention and describes the best mode contemplated, no matter how detailed the above appears in text, the invention can be practiced in many ways. Details of the system may vary considerably in implementation details, while still being encompassed by the invention disclosed herein. As noted above, particular terminology used when describing certain features or aspects of the invention should not be taken to imply that the terminology is being redefined herein to be restricted to any specific characteristics, features, or aspects of the invention with which that terminology is associated. In general, the terms used in the following claims should not be construed to limit the invention to the specific embodiments disclosed in the specification, unless the above Detailed Description section explicitly defines such terms. Accordingly, the actual scope of the invention encompasses not only the disclosed embodiments, but also all equivalent ways of practicing or implementing the invention under the claims.

While certain aspects of the invention are presented below in certain claim forms, the inventors contemplate the various aspects of the invention in any number of claim forms. For example, while only one aspect of the invention is recited as embodied in a computer-readable medium, other aspects may likewise be embodied in a computer-readable medium. Accordingly, the inventors reserve the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the invention.

We claim:

1. A non-transitory computer-readable storage medium storing instructions, which when executed by at least one computer, performs a method of managing users in a data management system that is configured to manage secondary copies of data files, the method comprising:

receiving a request from an identified preexisting user to perform a storage operation that would create a secondary copy of a particular production data file;

querying a security system to determine certain access rights of the identified preexisting user, wherein the certain access rights relate to the preexisting user's rights to access the particular production data file,

wherein querying the security system to determine the certain access rights includes determining one or more computers to which the identified preexisting user has access permission, and

US 8,447,728 B2

11

wherein the certain access rights permit the identified preexisting user to perform the requested storage operation if the particular production data file is associated with one of the determined one or more computers; and,

performing the requested storage operation to create a secondary copy of the particular production data file when the certain access rights permit the identified preexisting user to perform the requested storage operation,

wherein the secondary copies are useable to restore production data from which the secondary copies are created and wherein the secondary copies are not actively used by a live data server or other computer system; and,

wherein the certain access rights determine which copies of source data stored in multiple copies a user within a group can access.

2. The method of claim 1, further comprising permitting a user of the data management system, who does not have privileges to create new users within the data management system, to add the identified preexisting user to the group within the data management system.

3. The method of claim 1, wherein the particular production data file includes textual content, and the certain access rights are determined in part by evaluating the textual content.

4. The method of claim 1, further comprising:

querying the security system to determine an electronic address associated with the identified preexisting user; and,

notifying the identified preexisting user at the determined electronic address that the requested storage operation failed.

5. The method of claim 1, wherein performing the requested storage operation comprises creating a secondary copy of the particular production data file and applying access rights that the security system associates with the particular production data file to the created secondary copy.

6. The method of claim 1, further comprising querying the security system to determine an email address associated with the identified preexisting user.

7. The method of claim 1, further comprising:

identifying the preexisting user created in the security system, wherein the security system is external to the data management system, and wherein the identified preexisting user has the certain access rights defined by the security system;

creating the group within the data management system that associates one or more users with the at least one access right for performing storage operations that create secondary copies of data files from source production data files;

adding the identified preexisting user to the group within the data management system before receiving the request from the identified preexisting user to perform a storage operation.

8. A method for controlling a computer system to migrate users from a preexisting security system to a data management system that is configured to manage secondary copies of data files, wherein the secondary copies are useable to restore production data from which the secondary copies are created and are not actively used by a live data server or other computer system, by a method comprising:

migrating a selected preexisting security entity defined by an external security infrastructure to a new security entity in the data management system; and,

wherein migrating the selected preexisting security entity defined by the external security infrastructure

12

to the new security entity in the data management system includes associating the new security entity with a reference to the selected preexisting security entity in the security infrastructure; and

performing a storage operation requested by a selected preexisting security entity,

wherein the storage operation creates a secondary copy of a particular production data file,

wherein the storage operation is performed after a querying of the security infrastructure has been performed to determine that the selected preexisting security entity has sufficient access rights with respect to the particular production data file to perform the requested data management operation,

wherein querying the security infrastructure to determine that the selected preexisting security entity has sufficient access comprises determining one or more computers to which the selected preexisting security entity has access,

wherein the selected preexisting security entity has sufficient access rights when the particular production data file is associated with one of the determined one or more computers, and,

wherein the at least one privilege for performing storage management operations determines which copies of source data stored in multiple copies can be accessed by the new security entity.

9. The method of claim 8 wherein migrating the selected preexisting security entity further includes determining information about the preexisting security entity and associating at least a portion of the information with the new security entity.

10. The method of claim 8 wherein the external security infrastructure includes a directory provided by an operating system.

11. The method of claim 8 wherein providing a list of one or more preexisting security entities defined by a security infrastructure external to the data management system includes retrieving information from a first external security infrastructure provided by a first operating system.

12. The method of claim 8 wherein providing a list of one or more preexisting security entities defined by a security infrastructure external to the data management system further includes retrieving information from a second external security infrastructure provided by a second operating system.

13. The method of claim 8 wherein the selected preexisting security entity is an individual user.

14. The method of claim 8 wherein the external security infrastructure provides one or more access control lists that define one or more access rights assigned to each preexisting security entity.

15. The method of claim 8, further comprising:

receiving a request to create the new security entity in the data management system, wherein the new security entity is associated with at least one privilege for performing storage management operations to create secondary copies of data files from source production data files;

providing a list of one or more preexisting security entities defined by the security infrastructure external to the data management system; and,

receiving a selection of a preexisting security entity defined by the external security infrastructure.

16. A system for securing storage operations in a storage management system, wherein the storage management system interfaces with an external security component configured to store data regarding one or more external users and

US 8,447,728 B2

13

one or more access rights that indicate how the one or more external users are permitted to access production data files, the system comprising:

a memory;

means for managing data storage, wherein the means for managing data storage is configured to perform storage operations on behalf of one or more storage management users,

wherein some of the storage operations performed create secondary copies of data files from source production data files, and

wherein the secondary copies are useable to restore production data from which the secondary copies are created and are not actively used by a live data server or other computer system;

means for creating storage management users based on selected external users, and for determining whether a storage management user has sufficient access rights to perform a storage operation to create a secondary copy of a data file from a particular production data file, wherein this determination is made by querying the external security component to determine one or more access rights that indicate how the particular selected external user is permitted to access the particular production data file;

wherein determining whether a storage management user that was created based on a particular selected external user has sufficient access rights to perform a storage operation includes determining one or more computers to which the particular selected external user has access,

wherein the particular selected external user has sufficient access rights when the particular production data file is associated with one of the determined one or more computers;

wherein creating storage management users based on selected external users includes associating one or more storage management users with a reference to one or more of the selected external users; and

wherein the access rights to perform a storage operation determine which copies of source data stored in multiple copies a storage management user can access.

17. The system of claim **16** wherein the privileges migration component is further configured to determine at least a name and an email address associated with each of the selected external users.

14

18. The system of claim **16** wherein the storage management application is further configured to store, with each storage management user, privileges information describing storage management operations that each storage management user is allowed to perform.

19. The system of claim **16** wherein the external security component includes a Lightweight Directory Access Protocol (LDAP) directory.

20. One or more computer memories storing a data structure for associating access control information in a data management system with backup data, comprising:

a security descriptor having one or more access control lists,

wherein each access control list contains one or more access control entries,

wherein the access control entries contain users and groups defined by a security system that is external to the data management system,

wherein the access control list and the access control entities enable the users and the groups to have access rights to perform management storage operations with the backup data,

wherein the backup data is used, via the management storage operations, to restore production data from which the backup data is created and the backup data is not actively used by a live data server or other computer system,

wherein the one or more access control list include varying types of access control lists that provide the users and the groups with varying levels of security permissions; and

a backup data reference that specifies the backup data for which the security descriptor specifies access control information.

21. The computer memories of claim **20**, wherein the access control lists identify one or more computers to which the users and the groups have access, and

wherein the access control list permits the users and the groups to perform management storage operations on the one or more computers using the backup data if the backup data is associated with the one or more computers to which the users and the groups have access.

22. The computer memories of claim **20**, wherein the access rights to perform management storage operations with the backup data is based on content of or metadata associated with a data object in the backup data.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 8,447,728 B2
APPLICATION NO. : 13/250997
DATED : May 21, 2013
INVENTOR(S) : Anand Prahlad et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Specification

In column 2, line 54, delete “copied” and insert -- copied and --, therefor.

Signed and Sealed this
Third Day of September, 2013

A handwritten signature in cursive script, appearing to read "Teresa Stanek Rea".

Teresa Stanek Rea
Acting Director of the United States Patent and Trademark Office

EXHIBIT D

(12) **United States Patent**
Prahlad et al.

(10) **Patent No.:** **US 9,740,723 B2**
 (45) **Date of Patent:** **Aug. 22, 2017**

(54) **SYSTEMS AND METHODS FOR
 MANAGEMENT OF VIRTUALIZATION
 DATA**

(71) Applicant: **CommVault Systems, Inc.**, Oceanport,
 NJ (US)

(72) Inventors: **Anand Prahlad**, Bangalore (IN); **Rahul
 S. Pawar**, Marlboro, NJ (US); **Prakash
 Varadharajan**, Manalapan, NJ (US);
Pavan Kumar Reddy Bedadala,
 Piscataway, NJ (US)

(73) Assignee: **Commvault Systems, Inc.**, Tinton
 Falls, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this
 patent is extended or adjusted under 35
 U.S.C. 154(b) by 331 days.

(21) Appl. No.: **14/275,381**

(22) Filed: **May 12, 2014**

(65) **Prior Publication Data**

US 2014/0250093 A1 Sep. 4, 2014

Related U.S. Application Data

(62) Division of application No. 13/667,890, filed on Nov.
 2, 2012, now Pat. No. 8,725,973, which is a division
 (Continued)

(51) **Int. Cl.**
G06F 12/00 (2006.01)
G06F 13/00 (2006.01)
 (Continued)

(52) **U.S. Cl.**
 CPC **G06F 17/30336** (2013.01); **G06F 9/455**
 (2013.01); **G06F 11/1453** (2013.01);
 (Continued)

(58) **Field of Classification Search**
 CPC .. G06F 9/455; G06F 11/1453; G06F 11/1458;
 G06F 11/1469
 (Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,686,620 A 8/1987 Ng
 4,995,035 A 2/1991 Cole et al.
 (Continued)

FOREIGN PATENT DOCUMENTS

EP 0259912 A1 3/1988
 EP 0405926 A2 1/1991
 (Continued)

OTHER PUBLICATIONS

Commvault, "Automatic File System Multi-Streaming," [http://
 documentation.commvault.com/hds/release_7.0.0/books_online/
 1/enlight_us/feature](http://documentation.commvault.com/hds/release_7.0.0/books_online/1/enlight_us/feature), downloaded Jun. 4, 2015, 4 pages.

(Continued)

Primary Examiner — Charles Rones

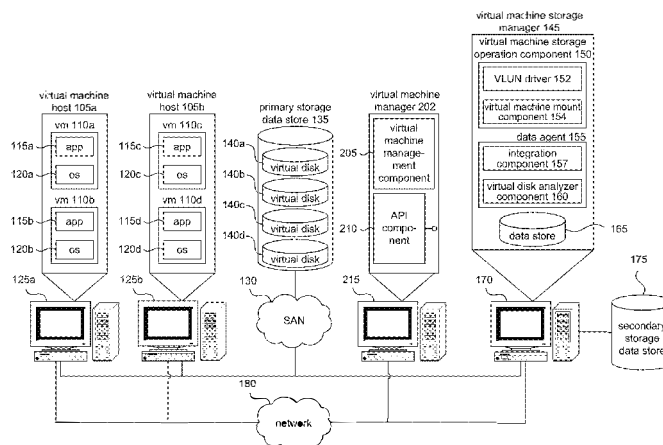
Assistant Examiner — Sidney Li

(74) *Attorney, Agent, or Firm* — Perkins Coie LLP

(57) **ABSTRACT**

Described in detail herein is a method of copying data of one or more virtual machines being hosted by one or more non-virtual machines. The method includes receiving an indication that specifies how to perform a copy of data of one or more virtual machines hosted by one or more virtual machine hosts. The method may include determining whether the one or more virtual machines are managed by a virtual machine manager that manages or facilitates management of the virtual machines. If so, the virtual machine manager is dynamically queried to automatically determine the virtual machines that it manages or that it facilitates management of. If not, a virtual machine host is dynamically queried to automatically determine the virtual machines that it hosts. The data of each virtual machine is then copied according to the specifications of the received indication.

21 Claims, 21 Drawing Sheets



US 9,740,723 B2

Page 2

Related U.S. Application Data

- of application No. 12/553,294, filed on Sep. 3, 2009, now Pat. No. 8,307,177.
- (60) Provisional application No. 61/169,515, filed on Apr. 15, 2009, provisional application No. 61/121,383, filed on Dec. 10, 2008, provisional application No. 61/094,753, filed on Sep. 5, 2008.
- (51) **Int. Cl.**
G06F 13/28 (2006.01)
G06F 17/30 (2006.01)
G06F 11/14 (2006.01)
G06F 9/455 (2006.01)
- (52) **U.S. Cl.**
 CPC **G06F 11/1458** (2013.01); **G06F 11/1469** (2013.01); **G06F 11/1456** (2013.01); **G06F 2201/815** (2013.01); **G06F 2201/84** (2013.01)
- (58) **Field of Classification Search**
 USPC 711/162
 See application file for complete search history.

(56) References Cited

U.S. PATENT DOCUMENTS

5,005,122 A 4/1991 Griffin et al.
 5,093,912 A 3/1992 Dong et al.
 5,133,065 A 7/1992 Cheffetz et al.
 5,193,154 A 3/1993 Kitajima et al.
 5,212,772 A 5/1993 Masters
 5,226,157 A 7/1993 Nakano et al.
 5,239,647 A 8/1993 Anglin et al.
 5,241,668 A 8/1993 Eastridge et al.
 5,241,670 A 8/1993 Eastridge et al.
 5,276,860 A 1/1994 Fortier et al.
 5,276,867 A 1/1994 Kenley et al.
 5,287,500 A 2/1994 Stoppani, Jr.
 5,321,816 A 6/1994 Rogan et al.
 5,333,315 A 7/1994 Saether et al.
 5,347,653 A 9/1994 Flynn et al.
 5,410,700 A 4/1995 Fecteau et al.
 5,448,724 A 9/1995 Hayashi
 5,491,810 A 2/1996 Allen
 5,495,607 A 2/1996 Pisello et al.
 5,504,873 A 4/1996 Martin et al.
 5,544,345 A 8/1996 Carpenter et al.
 5,544,347 A 8/1996 Yanai et al.
 5,559,957 A 9/1996 Balk
 5,619,644 A 4/1997 Crockett et al.
 5,638,509 A 6/1997 Dunphy et al.
 5,673,381 A 9/1997 Huai et al.
 5,699,361 A 12/1997 Ding et al.
 5,729,743 A 3/1998 Squibb
 5,751,997 A 5/1998 Kullick et al.
 5,758,359 A 5/1998 Saxon
 5,761,677 A 6/1998 Senator et al.
 5,764,972 A 6/1998 Crouse et al.
 5,778,395 A 7/1998 Whiting et al.
 5,812,398 A 9/1998 Nielsen
 5,813,009 A 9/1998 Johnson et al.
 5,813,017 A 9/1998 Morris
 5,875,478 A 2/1999 Blumenau
 5,887,134 A 3/1999 Ebrahim
 5,901,327 A 5/1999 Ofek
 5,924,102 A 7/1999 Perks
 5,950,205 A 9/1999 Aviani, Jr.
 5,974,563 A 10/1999 Beeler, Jr.
 6,021,415 A 2/2000 Cannon et al.
 6,026,414 A 2/2000 Anglin
 6,052,735 A 4/2000 Ulrich et al.
 6,076,148 A 6/2000 Kedem
 6,094,416 A 7/2000 Ying
 6,131,095 A 10/2000 Low et al.

6,131,190 A 10/2000 Sidwell
 6,148,412 A 11/2000 Cannon et al.
 6,154,787 A 11/2000 Urevig et al.
 6,161,111 A 12/2000 Mutalik et al.
 6,167,402 A 12/2000 Yeager
 6,212,512 B1 4/2001 Barney et al.
 6,260,069 B1 7/2001 Anglin
 6,269,431 B1 7/2001 Dunham
 6,275,953 B1 8/2001 Vahalia et al.
 6,301,592 B1 10/2001 Aoyama et al.
 6,324,581 B1 11/2001 Xu et al.
 6,328,766 B1 12/2001 Long
 6,330,570 B1 12/2001 Crighton
 6,330,642 B1 12/2001 Carteau
 6,343,324 B1 1/2002 Hubis et al.
 RE37,601 E 3/2002 Eastridge et al.
 6,356,801 B1 3/2002 Goodman et al.
 6,389,432 B1 5/2002 Pothapragada et al.
 6,421,711 B1 7/2002 Blumenau et al.
 6,487,561 B1 11/2002 Ofek et al.
 6,519,679 B2 2/2003 Deviredy et al.
 6,538,669 B1 3/2003 Lagueux, Jr. et al.
 6,564,228 B1 5/2003 O'Connor
 6,581,076 B1 6/2003 Ching et al.
 6,658,526 B2 12/2003 Nguyen et al.
 7,035,880 B1 4/2006 Crescenti et al.
 7,076,270 B2 7/2006 Jagers et al.
 7,219,162 B2 5/2007 Donker et al.
 7,246,207 B2 7/2007 Kottomtharayil et al.
 7,395,282 B1 7/2008 Crescenti et al.
 7,448,079 B2 11/2008 Tremain
 7,631,351 B2 12/2009 Erofeev
 7,730,035 B2 6/2010 Berger et al.
 7,756,835 B2 7/2010 Pugh
 7,756,964 B2 7/2010 Madison, Jr. et al.
 7,840,537 B2 11/2010 Gokhale et al.
 7,882,077 B2 2/2011 Gokhale et al.
 7,899,788 B2 3/2011 Chandhok et al.
 7,937,421 B2 5/2011 Mikesell et al.
 8,069,271 B2 11/2011 Brunet et al.
 8,140,786 B2 3/2012 Bunte et al.
 8,219,524 B2 7/2012 Gokhale
 8,307,177 B2 11/2012 Prahlad et al.
 8,396,838 B2 3/2013 Brockway et al.
 8,407,190 B2 3/2013 Prahlad et al.
 8,433,679 B2 4/2013 Crescenti et al.
 8,434,131 B2 4/2013 Varadharajan et al.
 8,473,594 B2 6/2013 Astete et al.
 8,620,870 B2 12/2013 Dwarampudi et al.
 2003/0037211 A1* 2/2003 Winokur G06F 11/1456 711/162
 2005/006704 A1* 3/2005 Bulson G06F 9/5077 718/1
 2005/0108709 A1* 5/2005 Sciandra G06F 9/505 718/1
 2006/0224846 A1 10/2006 Amarendran et al.
 2007/0203938 A1 8/2007 Prahlad et al.
 2007/0234302 A1 10/2007 Suzuki et al.
 2008/0028408 A1* 1/2008 Day G06F 9/5077 718/104
 2008/0134175 A1* 6/2008 Fitzgerald G06F 9/45533 718/1
 2009/0144416 A1 6/2009 Chatley et al.
 2009/0319534 A1 12/2009 Gokhale
 2011/0087632 A1 4/2011 Subramanian et al.
 2012/0254364 A1 10/2012 Vijayan
 2013/0061014 A1 3/2013 Prahlad et al.
 2013/0262390 A1 10/2013 Kumarasamy et al.
 2013/0262638 A1 10/2013 Kumarasamy et al.
 2013/0262801 A1 10/2013 Sancheti et al.
 2013/0290267 A1 10/2013 Dwarampudi et al.

FOREIGN PATENT DOCUMENTS

EP 0467546 A2 1/1992
 EP 0774715 A1 5/1997
 EP 0809184 A1 11/1997
 EP 0899662 A1 3/1999

US 9,740,723 B2

Page 3

(56)

References Cited

FOREIGN PATENT DOCUMENTS

EP	0981090	A1	2/2000
WO	9513580	A1	5/1995
WO	9912098	A1	3/1999

OTHER PUBLICATIONS

Hitachi, "Create a Virtual Machine—VM Lifecycle Management—Vmware," <http://documentation.commvault.com/hds/v10/article?p=products/vs/vmware/vm%20provisio...>, downloaded Apr. 28, 2015, 2 pages.

Hitachi, "Frequently Asked Questions—Virtual Server Agent for Vmware," <http://documentation.commvault.com/hds/v10/article?p=products/vs/vmware/faqs.htm>, downloaded Apr. 28, 2015, 11 pages.

Hitachi, "Overview—Virtual Server Agent for Vmware," <http://documentation.commvault.com/hds/v10/article?p=products/vs/vmware/overview.htm>, downloaded Apr. 28, 2015, 3 pages.

Hitachi, "Recover Virtual Machines or VM Files—Web Console," <http://documentation.commvault.com/hds/v10/article?p=products/vs/vmware/vm%20archivin...>, downloaded Apr. 28, 2015, 2 pages.

U.S. Appl. No. 13/765,389, filed Feb. 12, 2013, Kripalani.

Armstead et al., "Implementation of a Campwide Distributed Mass Storage Service: The Dream vs. Reality," IEEE, Sep. 11-14, 1995, pp. 190-199.

Arneson, "Mass Storage Archiving in Network Environments," Digest of Papers, Ninth IEEE Symposium on Mass Storage Systems, Oct. 31-Nov. 3, 1988, pp. 45-50, Monterey, CA.

Brandon, J., "Virtualization Shakes Up Backup Strategy," <http://www.computerworld.com>, internet accessed on Mar. 6, 2008, 3 pages.

Cabrera et al., "ADSM: A Multi-Platform, Scalable, Backup and Archive Mass Storage System," Digest of Papers, Compcon '95, Proceedings of the 40th IEEE Computer Society International Conference, Mar. 5-9, 1995, pp. 420-427, San Francisco, CA.

CommVault Systems, Inc., "A CommVault White Paper: VMware Consolidated Backup (VCB) Certification Information Kit," 2007, 23 pages.

CommVault Systems, Inc., "CommVault Solutions—Vmware," <http://www.commvault.com/solutions/vmware/>, internet accessed Mar. 24, 2008, 2 pages.

CommVault Systems, Inc., "Enhanced Protection and Manageability of Virtual Servers," Partner Solution Brief, 2008, 6 pages.

Davis, D., "3 VMware Consolidated Backup (VCB) Utilities You Should Know," Petri IT Knowledgebase, <http://www.petri.co.il/vmware-consolidated-backup-utilities.htm>, internet accessed on Jul. 14, 2008, 7 pages.

Davis, D., "Understanding VMware VMX Configuration Files," Petri IT Knowledgebase, http://www.petri.co.il/virtual_vmware_vmx_configuration_files.htm, internet accessed on Jun. 19, 2008, 6 pages.

Davis, D., "VMware Server & Workstation Disk Files Explained," Petri IT Knowledgebase, http://www.petri.co.il/virtual_vmware_files_explained.htm, internet accessed on Jun. 19, 2008, 5 pages.

Davis, D., "VMware Versions Compared," Petri IT Knowledgebase, http://www.petri.co.il/virtual_vmware_versions_compared.htm, internet accessed on Apr. 28, 2008, 6 pages.

Eitel, "Backup and Storage Management in Distributed Heterogeneous Environments," IEEE, Jun. 12-16, 1994, pp. 124-126.

Gait, J., "The Optical File Cabinet: A Random-Access File System for Write-Once Optical Disks," IEEE Computer, vol. 21, No. 6, pp. 11-22 (Jun. 1988).

International Preliminary Report on Patentability and Written Opinion for PCT/US2011/054374, dated Apr. 11, 2013, 6 pages.

International Search Report and Written Opinion for PCT/US2011/054374, dated May 2, 2012, 9 pages.

Jander, M., "Launching Storage-Area Net," Data Communications, US, McGraw Hill, NY, vol. 27, No. 4 (Mar. 21, 1998), pp. 64-72.

Microsoft Corporation, "How NTFS Works," Windows Server TechCenter, updated Mar. 28, 2003, internet accessed Mar. 26, 2008, 26 pages.

Rosenblum et al., "The Design and Implementation of a Log-Structured File System," Operating Systems Review SIGOPS, vol. 25, No. 5, New York, US, pp. 1-15 (May 1991).

Sanbarrow.com, "Disktype-table," <http://sanbarrow.com/vmdk/disktypes.html>, internet accessed on Jul. 22, 2008, 4 pages.

Sanbarrow.com, "Files Used by a VM," <http://sanbarrow.com/vmx/vmx-files-used-by-a-vm.html>, internet accessed on Jul. 22, 2008, 2 pages.

Sanbarrow.com, "Monolithic Versus Split Disks," <http://sanbarrow.com/vmdk/monolithicversussplit.html>, internet accessed on Jul. 14, 2008, 2 pages.

VMware, Inc., "Open Virtual Machine Format," <http://www.vmware.com/appliances/learn/ovf.html>, internet accessed on May 6, 2008, 2 pages.

VMware, Inc., "OVF, Open Virtual Machine Format Specification, version 0.9," White Paper, <http://www.vmware.com>, 2007, 50 pages.

VMware, Inc., "The Open Virtual Machine Format Whitepaper for OVF Specification, version 0.9," White Paper, <http://www.vmware.com>, 2007, 16 pages.

VMware, Inc., "Understanding VMware Consolidated Backup," White Paper, <http://www.vmware.com>, 2007, 11 pages.

VMware, Inc., "Using VMware Infrastructure for Backup and Restore," Best Practices, <http://www.vmware.com>, 2006, 20 pages.

VMware, Inc., "Virtual Disk API Programming Guide," <http://www.vmware.com>, Revision 20080411, 2008, 44 pages.

VMware, Inc., "Virtual Disk Format 1.1," VMware Technical Note, <http://www.vmware.com>, Revision 20071113, Version 1.1, 2007, 18 pages.

VMware, Inc., "Virtual Machine Backup Guide, ESX Server 3.0.1 and VirtualCenter 2.0.1," <http://www.vmware.com>, updated Nov. 21, 2007, 74 pages.

VMware, Inc., "Virtual Machine Backup Guide, ESX Server 3.5, ESX Server 3i version 3.5, VirtualCenter 2.5," <http://www.vmware.com>, updated Feb. 21, 2008, 78 pages.

VMware, Inc., "Virtualized iSCSI SANs: Flexible, Scalable Enterprise Storage for Virtual Infrastructures," White Paper, <http://www.vmware.com>, Mar. 2008, 13 pages.

VMware, Inc., "VMware Consolidated Backup, Improvements in Version 3.5," Information Guide, <http://www.vmware.com>, 2007, 11 pages.

VMware, Inc., "VMware Consolidated Backup," Product Datasheet, <http://www.vmware.com>, 2007, 2 pages.

VMware, Inc., "VMware ESX 3.5," Product Datasheet, <http://www.vmware.com>, 2008, 4 pages.

VMware, Inc., "VMware GSX Server 3.2, Disk Types: Virtual and Physical," http://www.vmware.com/support/gsx3/doc/disk_types_gsx.html, internet accessed on Mar. 25, 2008, 2 pages.

VMware, Inc., "VMware OVF Tool," Technical Note, <http://www.vmware.com>, 2007, 4 pages.

VMware, Inc., "VMware Workstation 5.0, Snapshots in a Linear Process," http://www.vmware.com/support/ws5/doc/ws_preserve_sshot_linear.html, internet accessed on Mar. 25, 2008, 1 page.

VMware, Inc., "VMware Workstation 5.0, Snapshots in a Process Tree," http://www.vmware.com/support/ws5/doc/ws_preserve_sshot_tree.html, internet accessed on Mar. 25, 2008, 1 page.

VMware, Inc., "VMware Workstation 5.5, What Files Make Up a Virtual Machine?" http://www.vmware.com/support/ws55/doc/ws_learning_files_in_a_vm.html, internet accessed on Mar. 25, 2008, 2 pages.

Wikipedia, "Cluster (file system)," http://en.wikipedia.org/wiki/Cluster_%28file_system%29, internet accessed Jul. 25, 2008, 1 page.

Wikipedia, "Cylinder-head-sector," <http://en.wikipedia.org/wiki/Cylinder-head-sector>, internet accessed Jul. 22, 2008, 6 pages.

Wikipedia, "File Allocation Table," http://en.wikipedia.org/wiki/File_Allocation_Table, internet accessed on Jul. 25, 2008, 19 pages.

US 9,740,723 B2

Page 4

(56)

References Cited

OTHER PUBLICATIONS

Wikipedia, "Logical Disk Manager," <http://en.wikipedia.org/wiki/Logical_Disk_Manager>, internet accessed Mar. 26, 2008, 3 pages.

Wikipedia, "Logical Volume Management," <http://en.wikipedia.org/wiki/Logical_volume_management>, internet accessed on Mar. 26, 2008, 5 pages.

Wikipedia, "Storage Area Network," <http://en.wikipedia.org/wiki/Storage_area_network>, internet accessed on Oct. 24, 2008, 5 pages.

Wikipedia, "Virtualization," <[http://en.wikipedia.org/wikiNirtualization](http://en.wikipedia.org/wiki/Nirtualization)>, internet accessed Mar. 18, 2008, 7 pages.

* cited by examiner

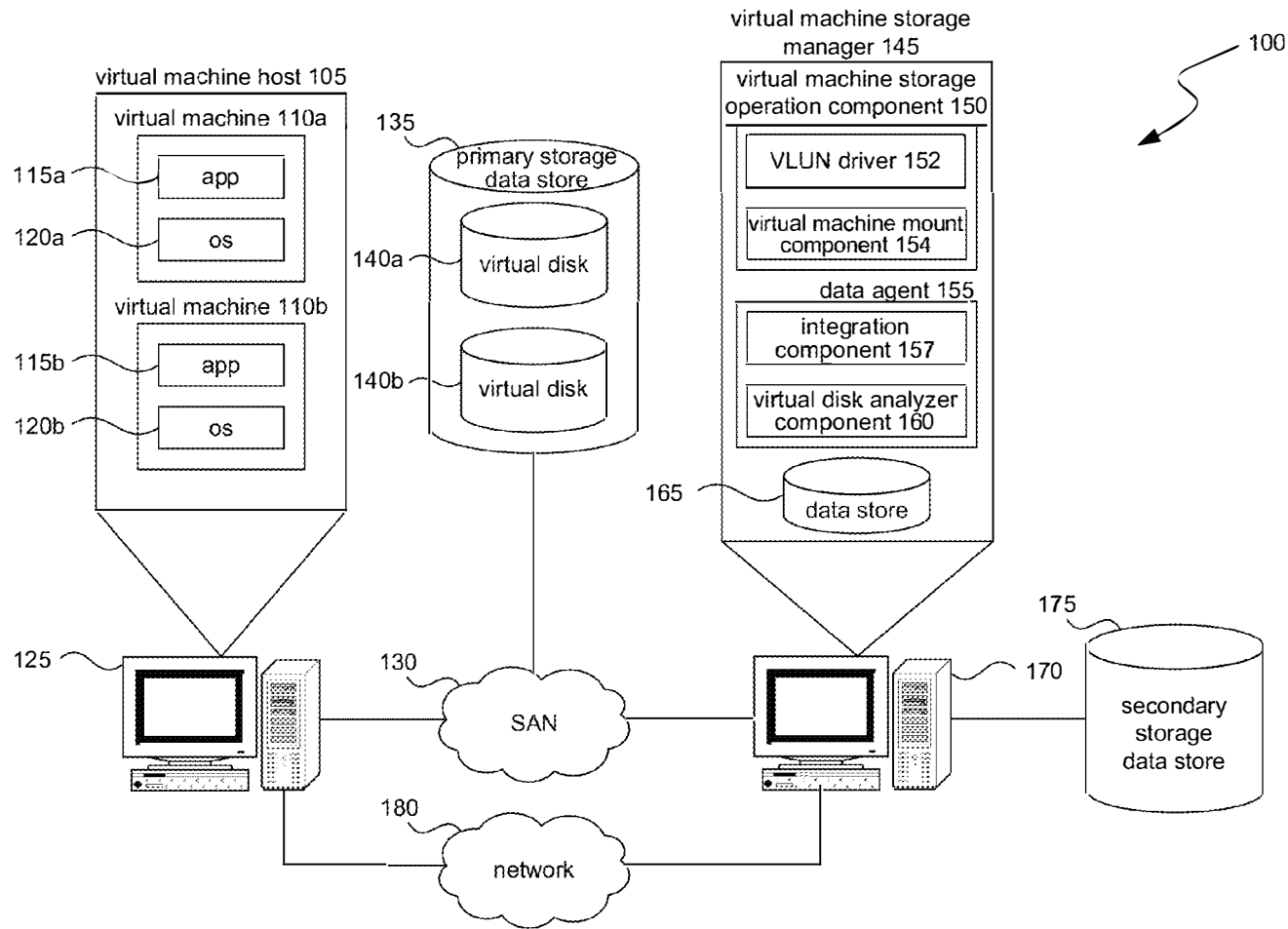


FIG. 1A

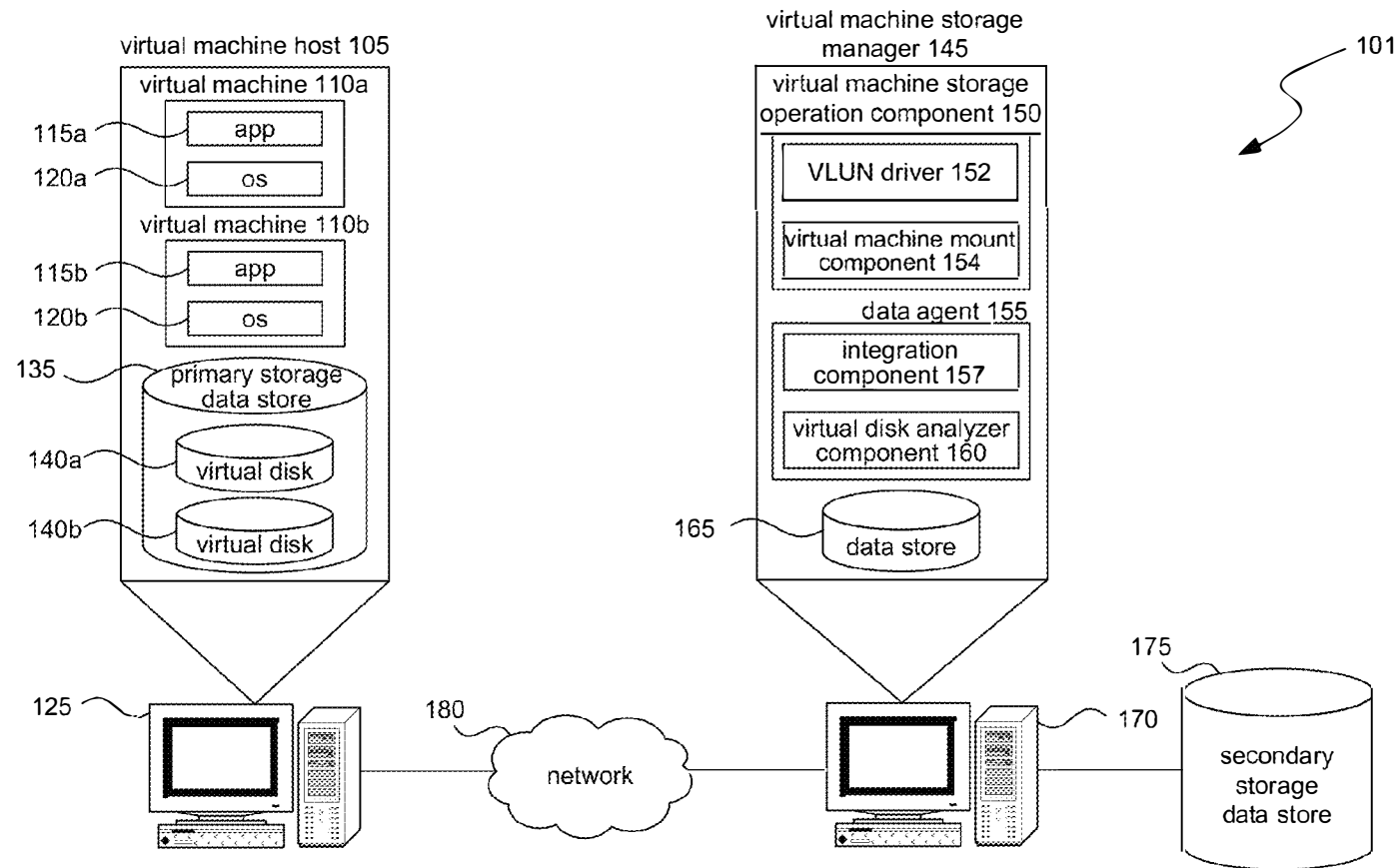


FIG. 1B

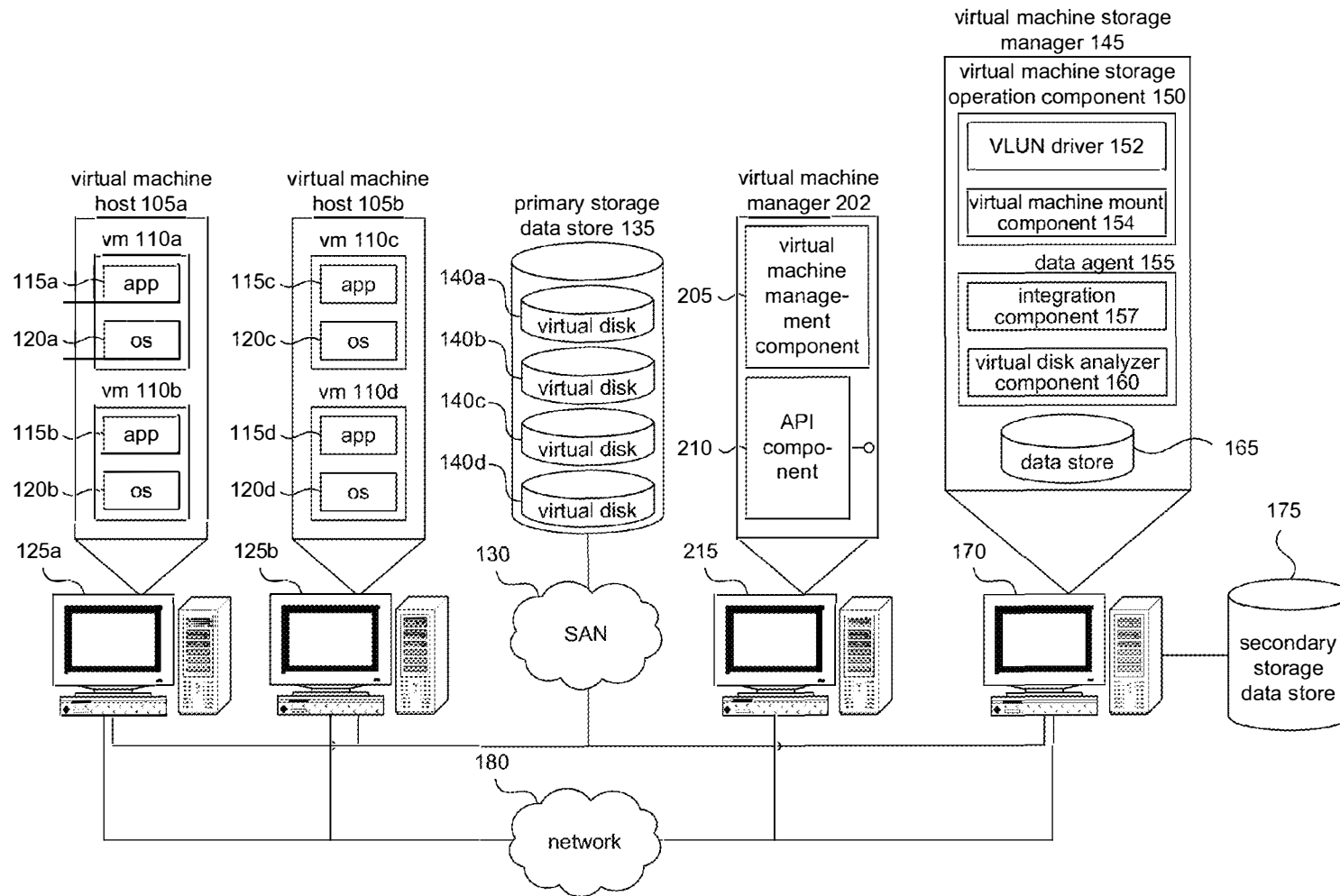
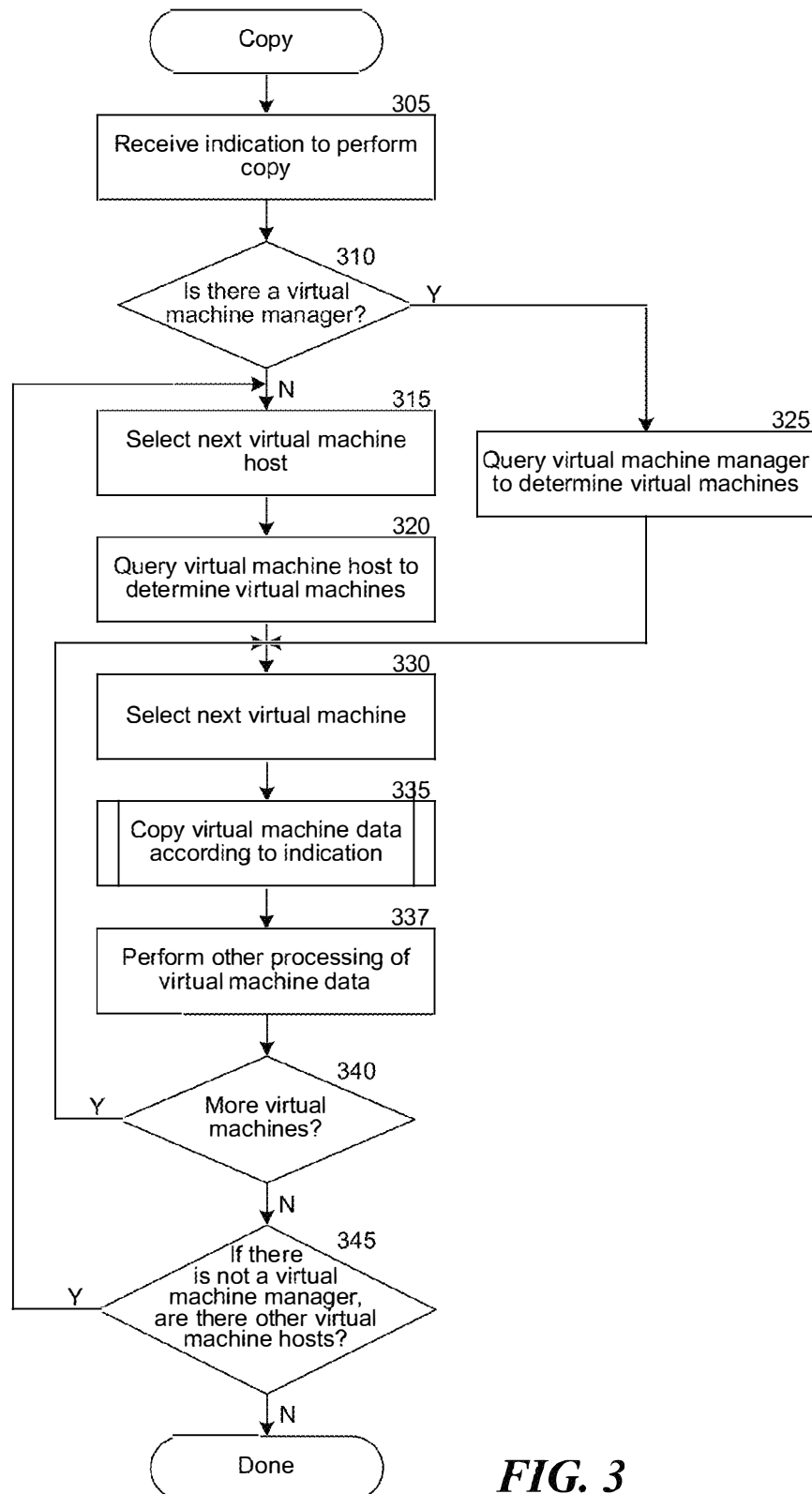


FIG. 2

**FIG. 3**

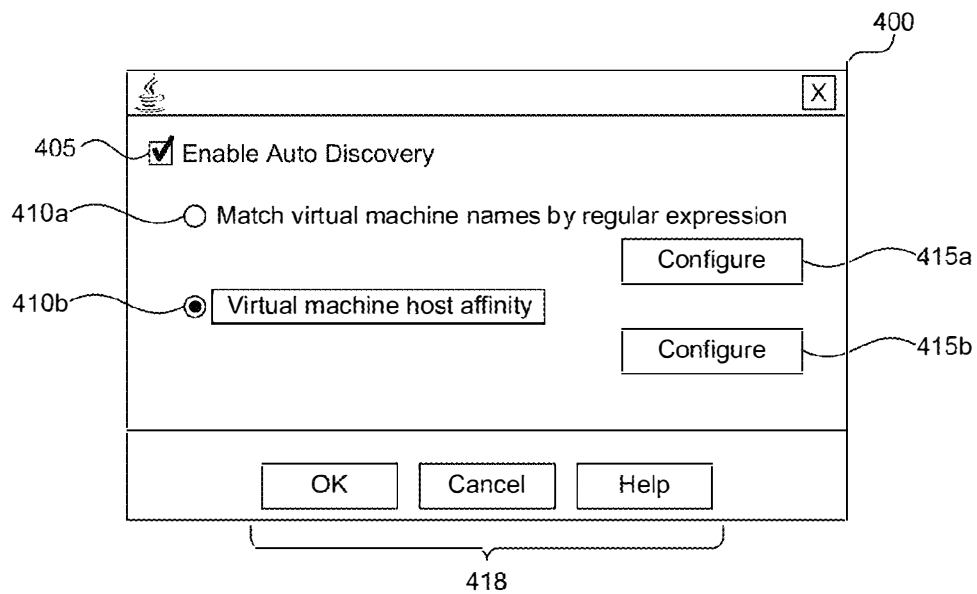
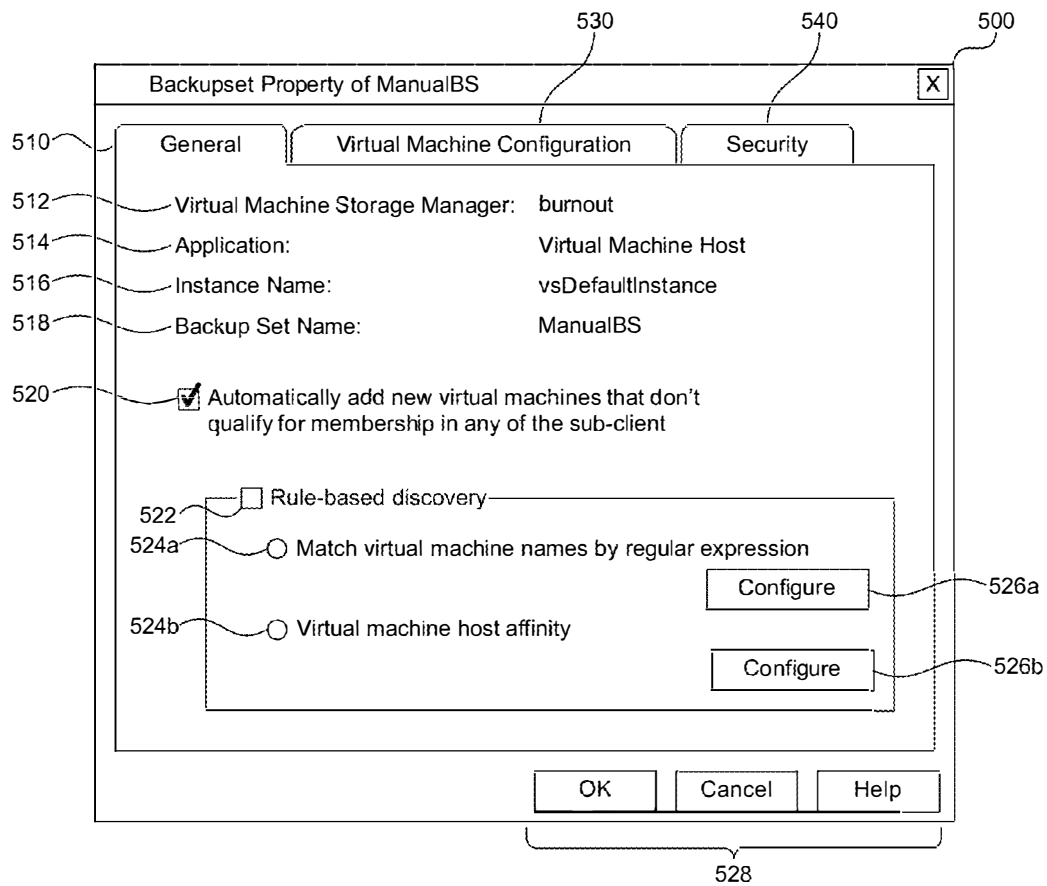


FIG. 4

**FIG. 5A**

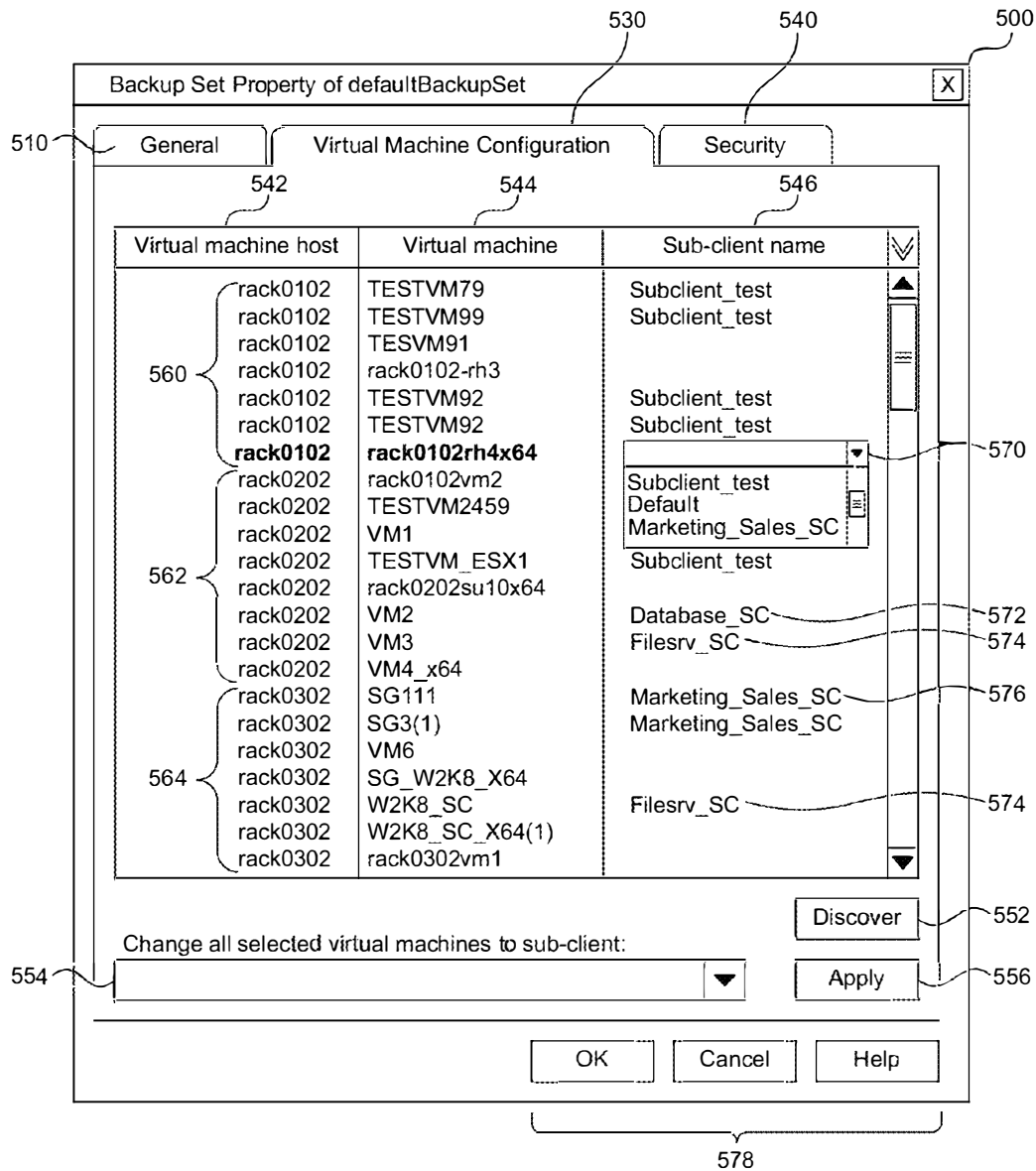


FIG. 5B

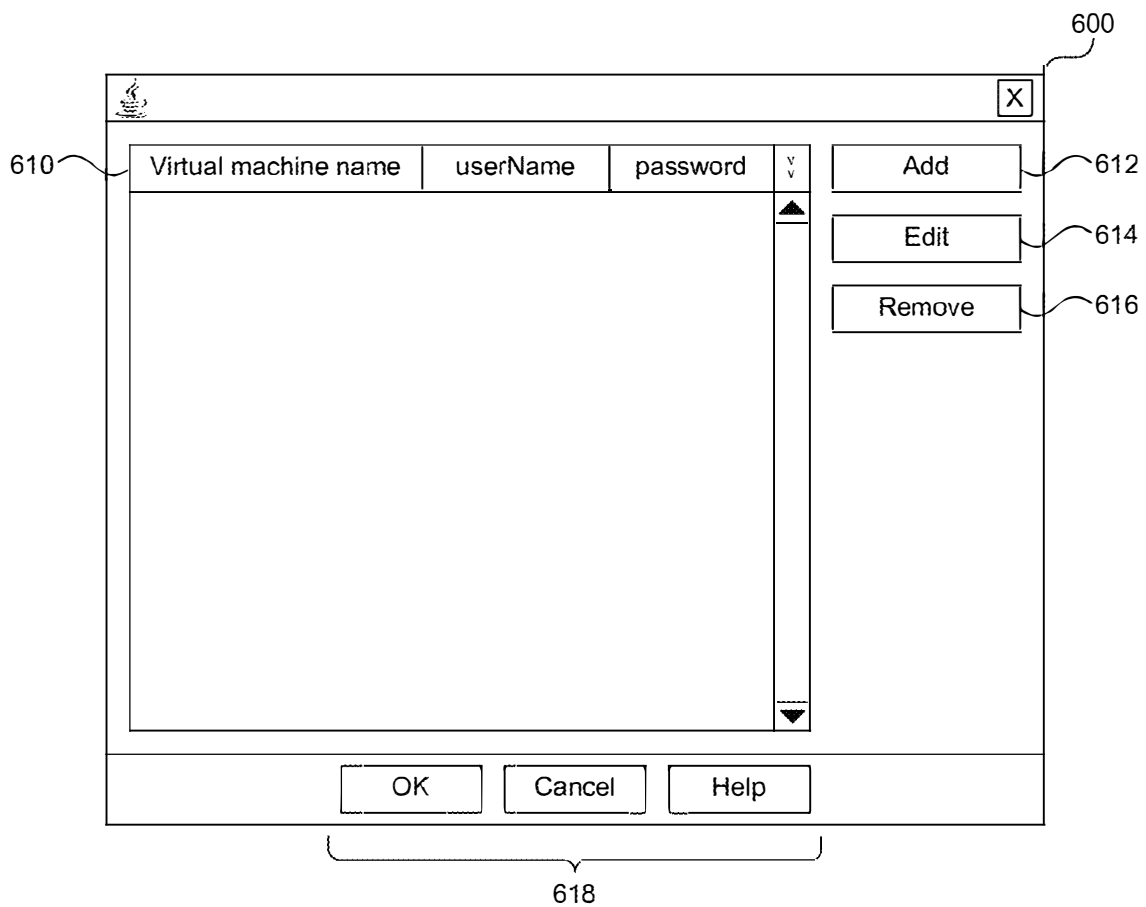


FIG. 6

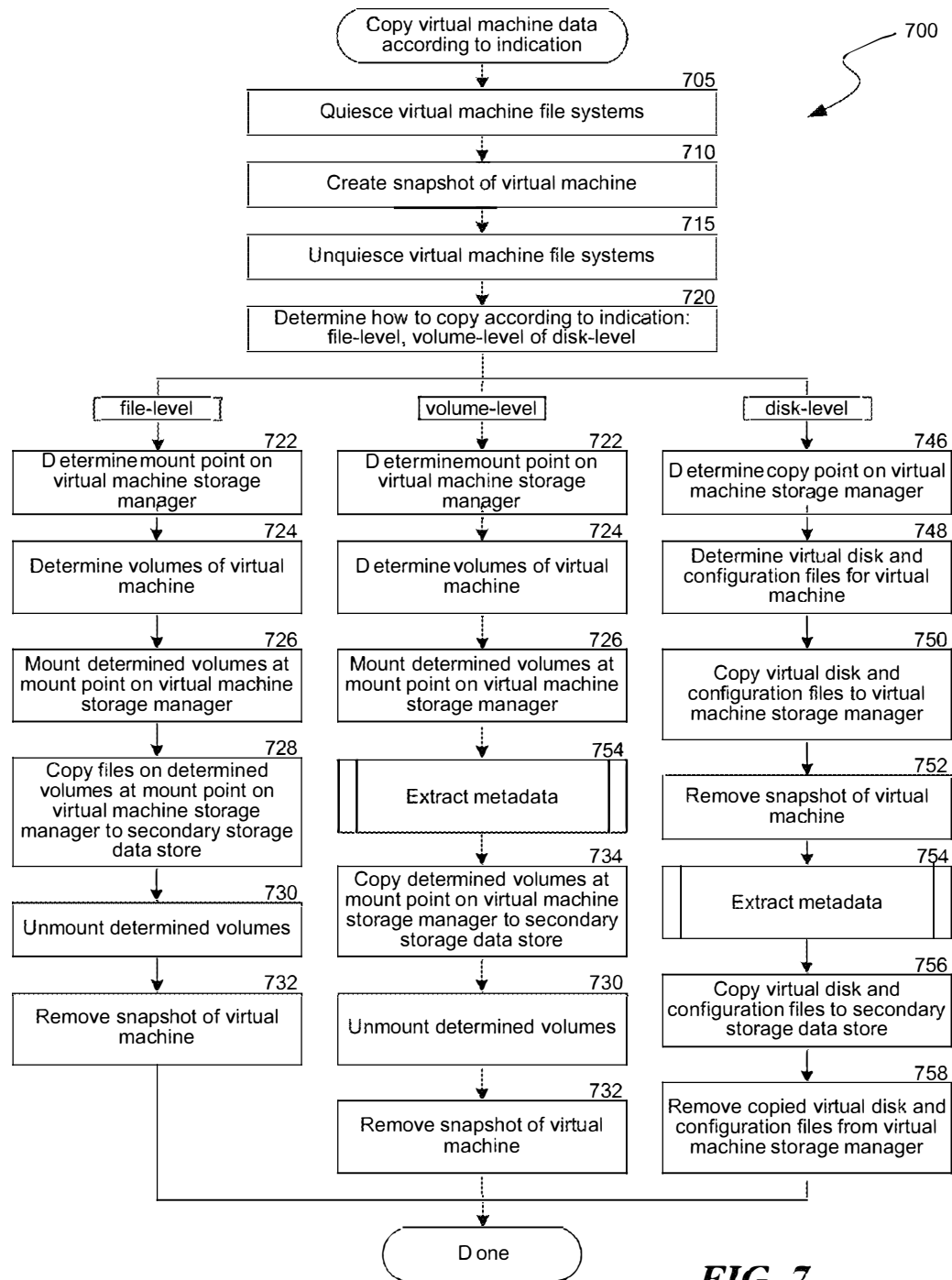


FIG. 7

U.S. Patent

Aug. 22, 2017

Sheet 10 of 21

US 9,740,723 B2

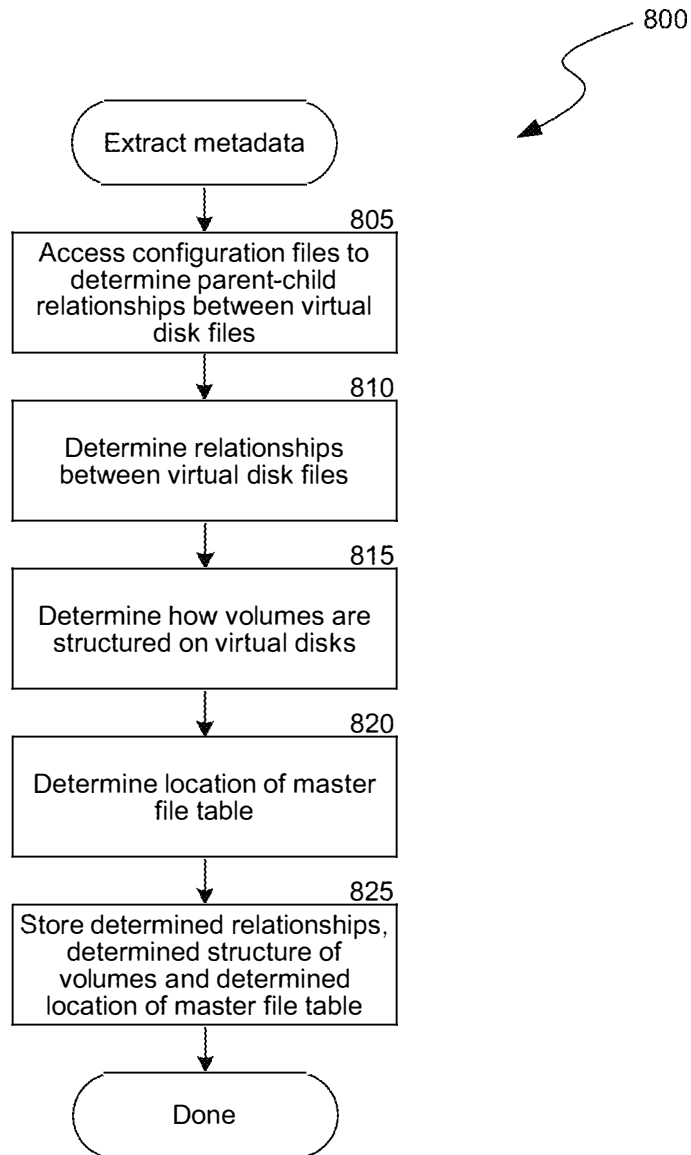


FIG. 8

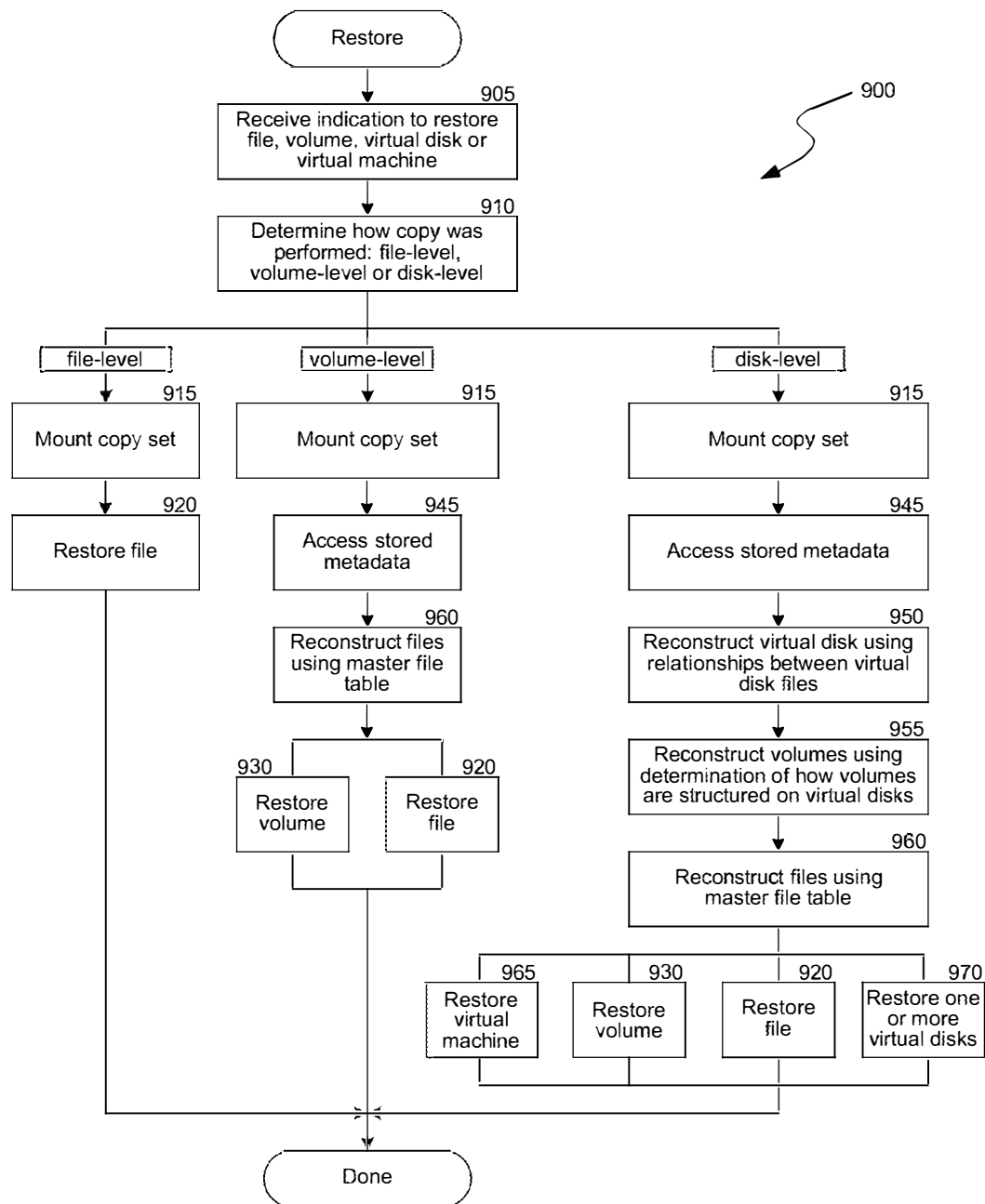
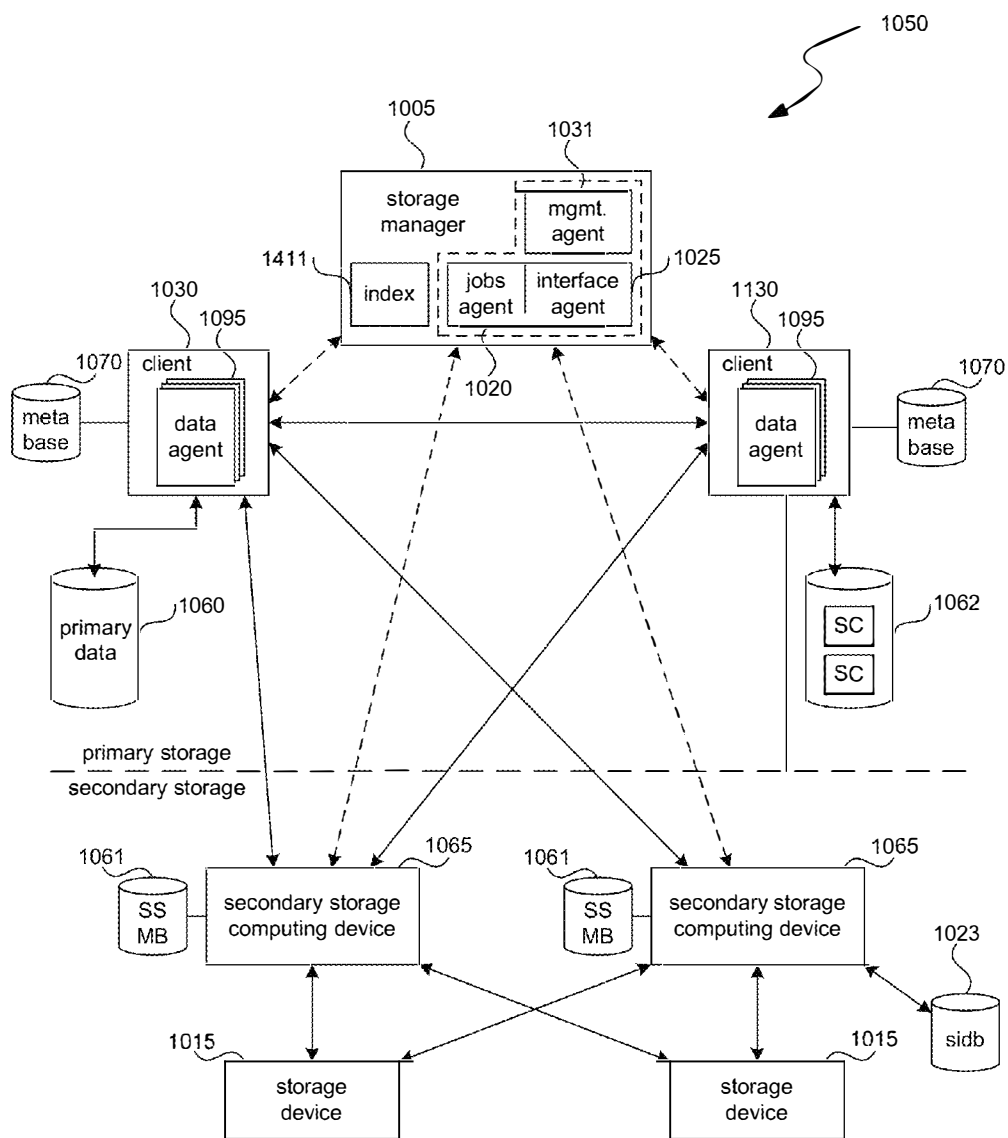


FIG. 9

**FIG. 10**

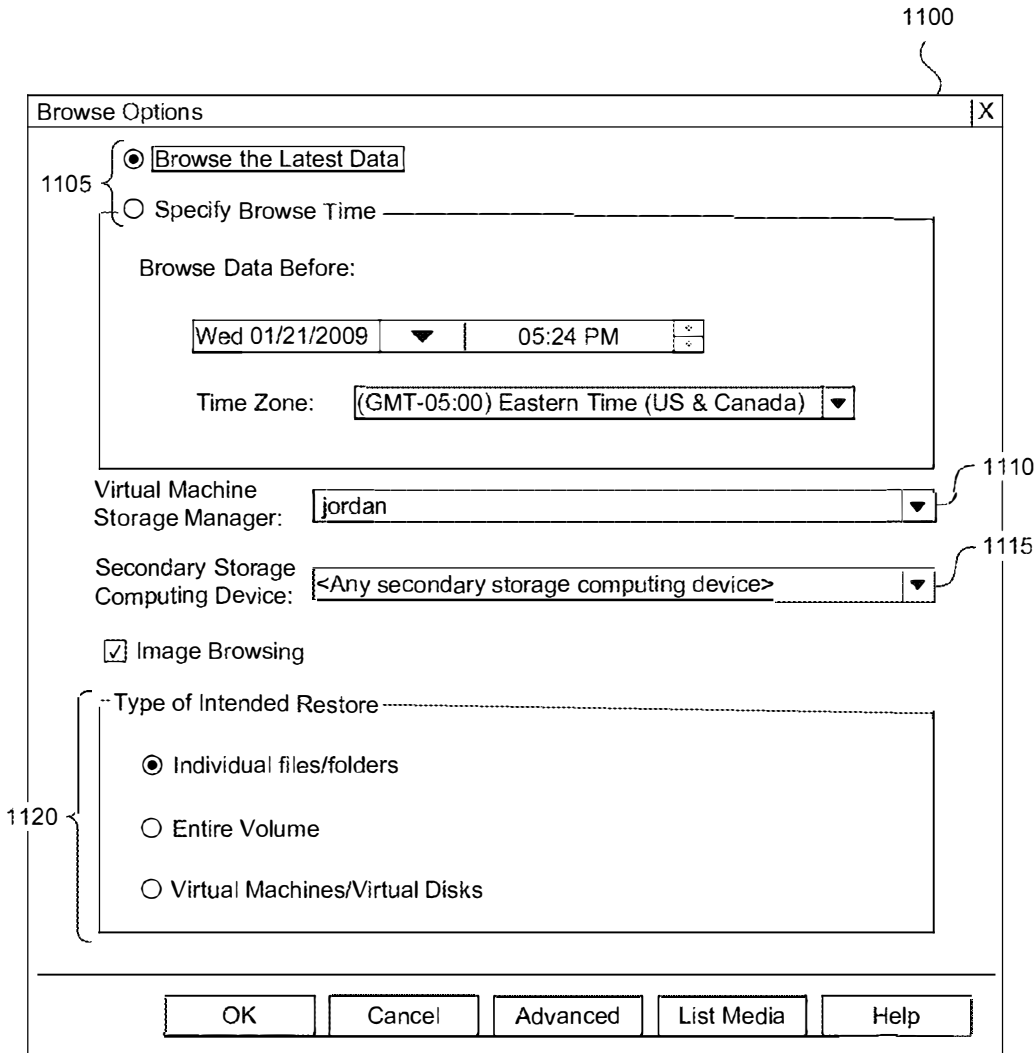


FIG. 11

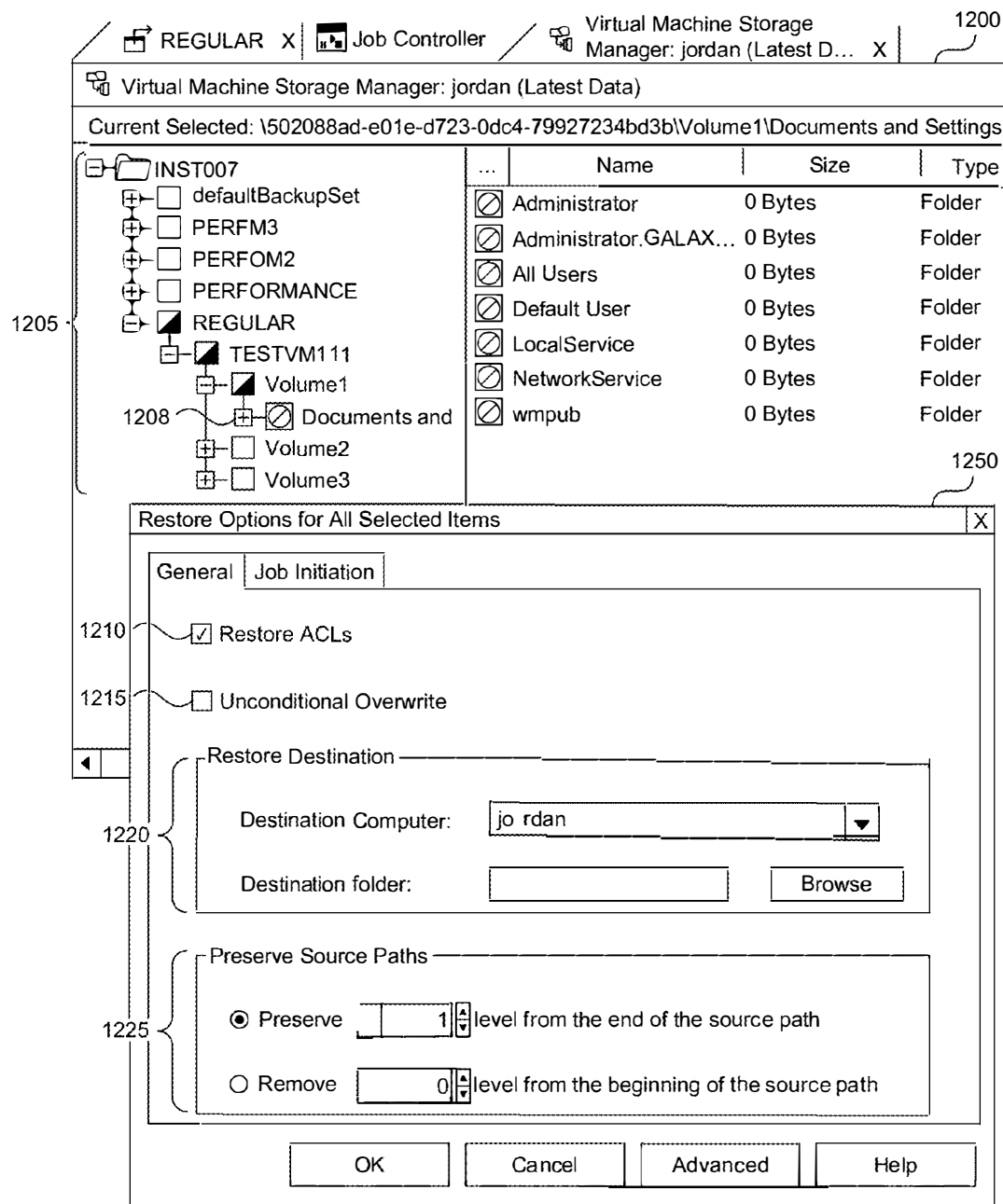


FIG. 12

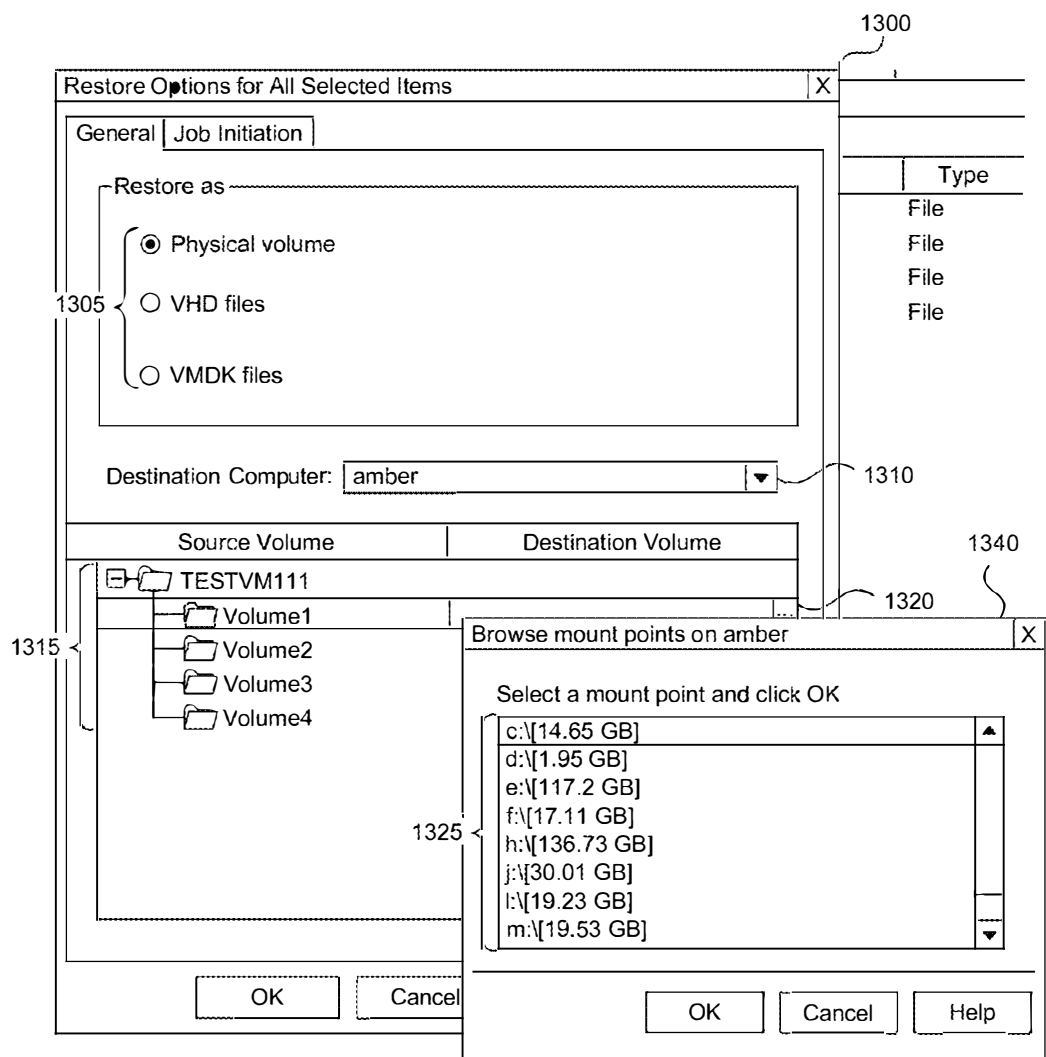


FIG. 13A

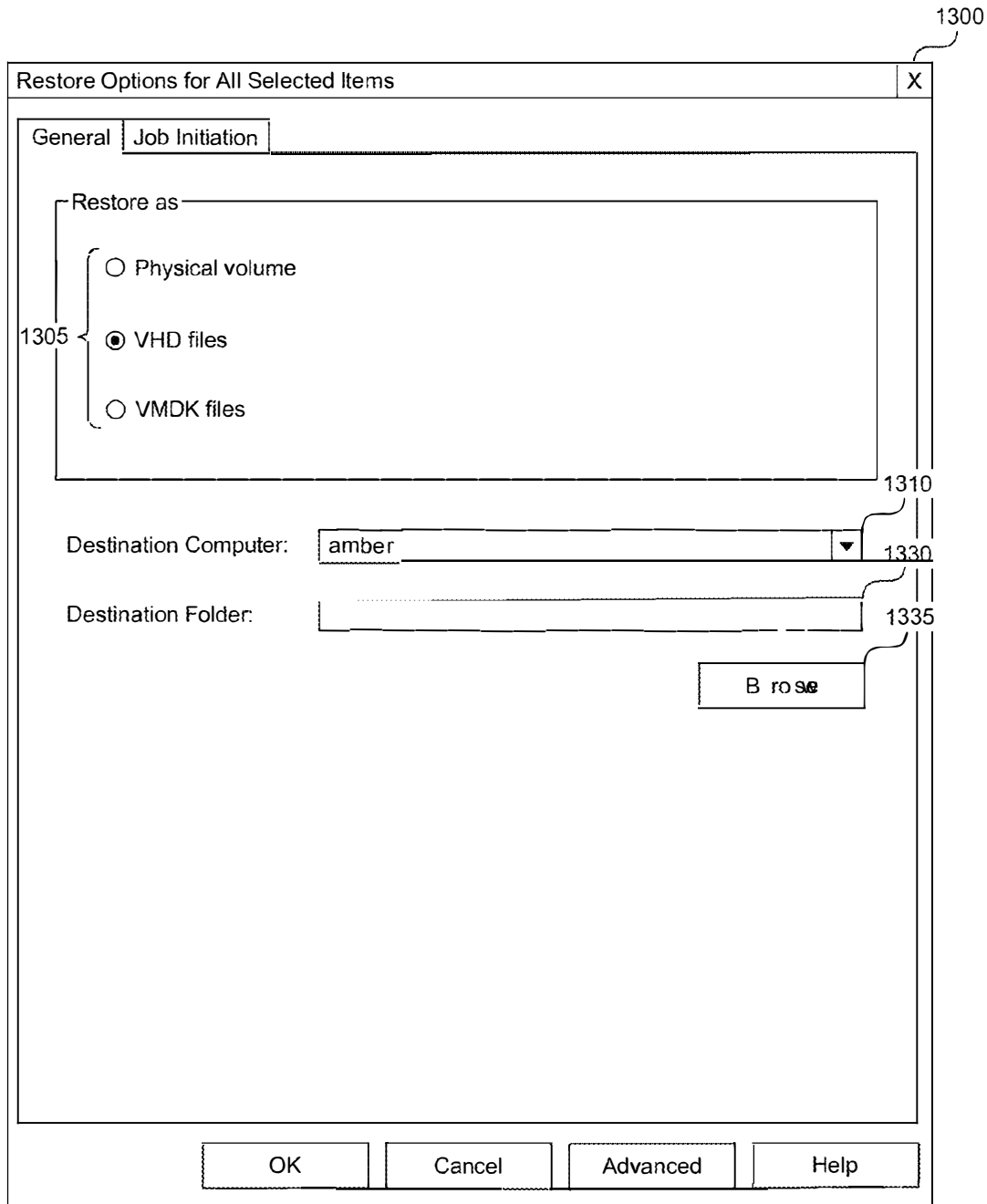


FIG. 13B

Restore Options for All Selected Items

General Job Initiation

Restore as

☒ Virtual Machines

☐ Virtual Disks

Destination Computer: jordan

Destination Folder: D:\MountFolder

Browse

Virtual Machine Restore Options

Virtual Machine Name TESTVM100

Server Name VirtualCenterServer.com

☒ Virtual Machine Manager

Virtual Machine Host managed by Virtual Machine Manager ESXServer.commvault.c

☐ Virtual Machine Host

Authentication

User Name: root

Password: *****

Confirm Password: *****

OK Cancel Advanced Help

FIG. 14A

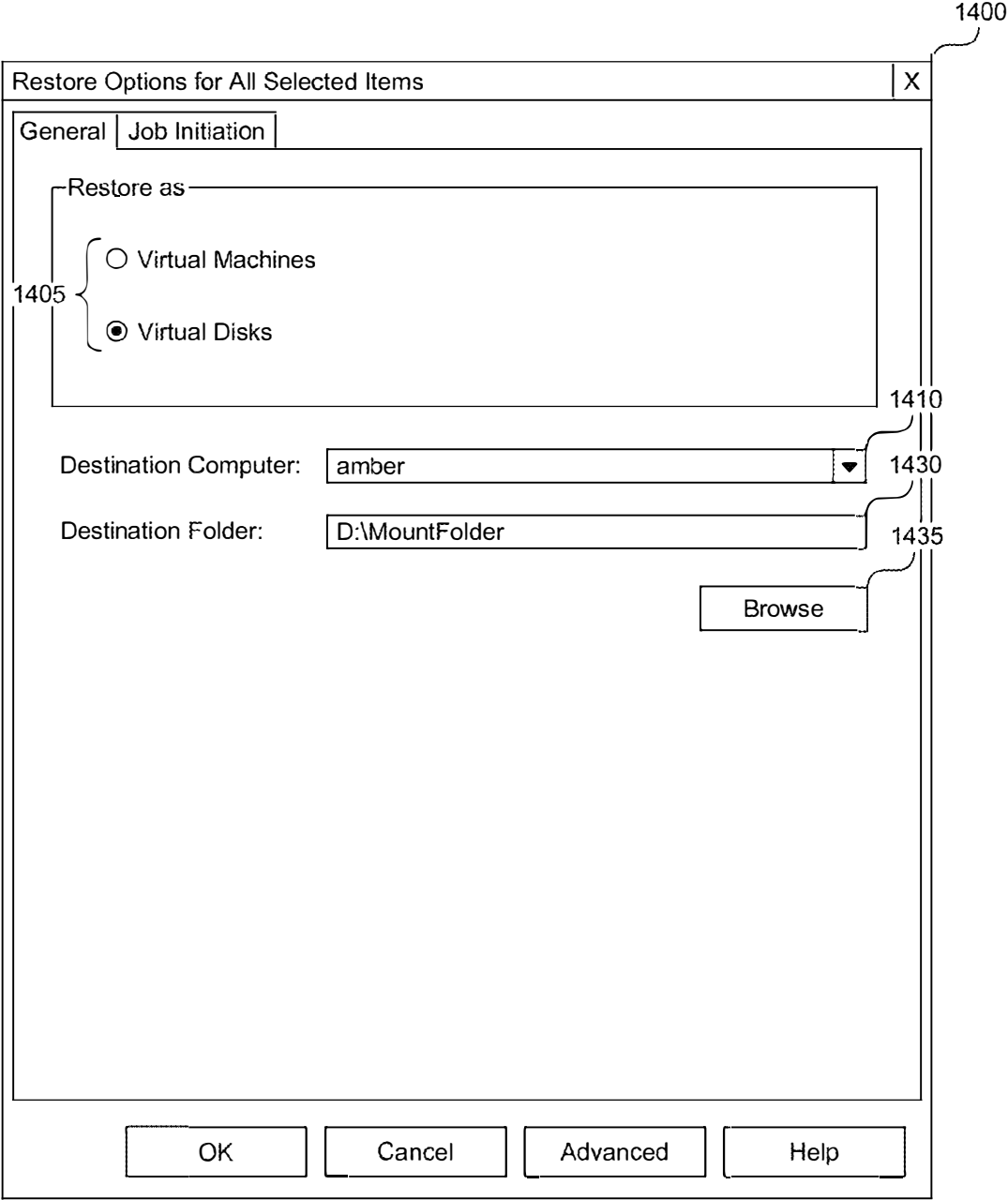


FIG. 14B

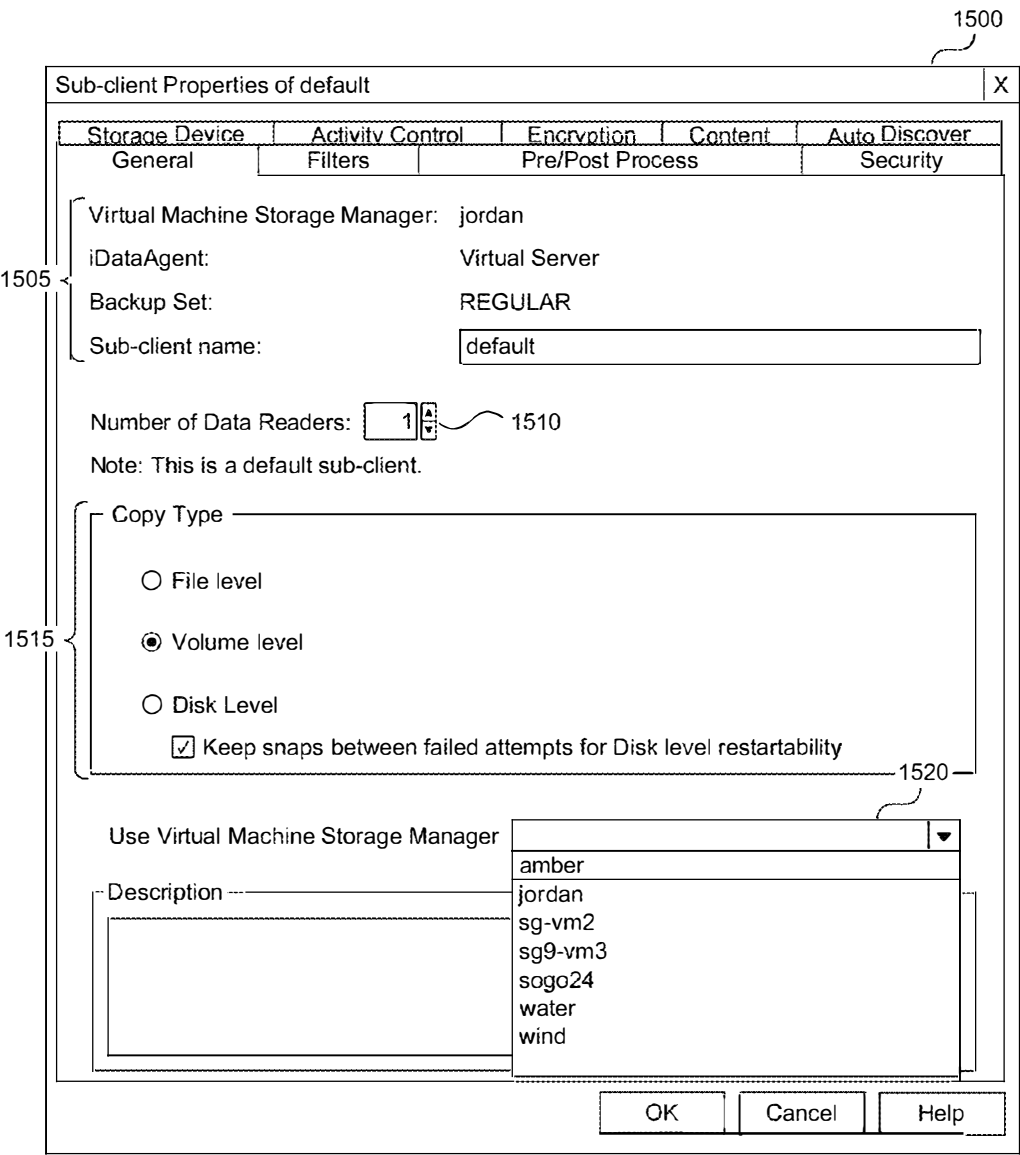
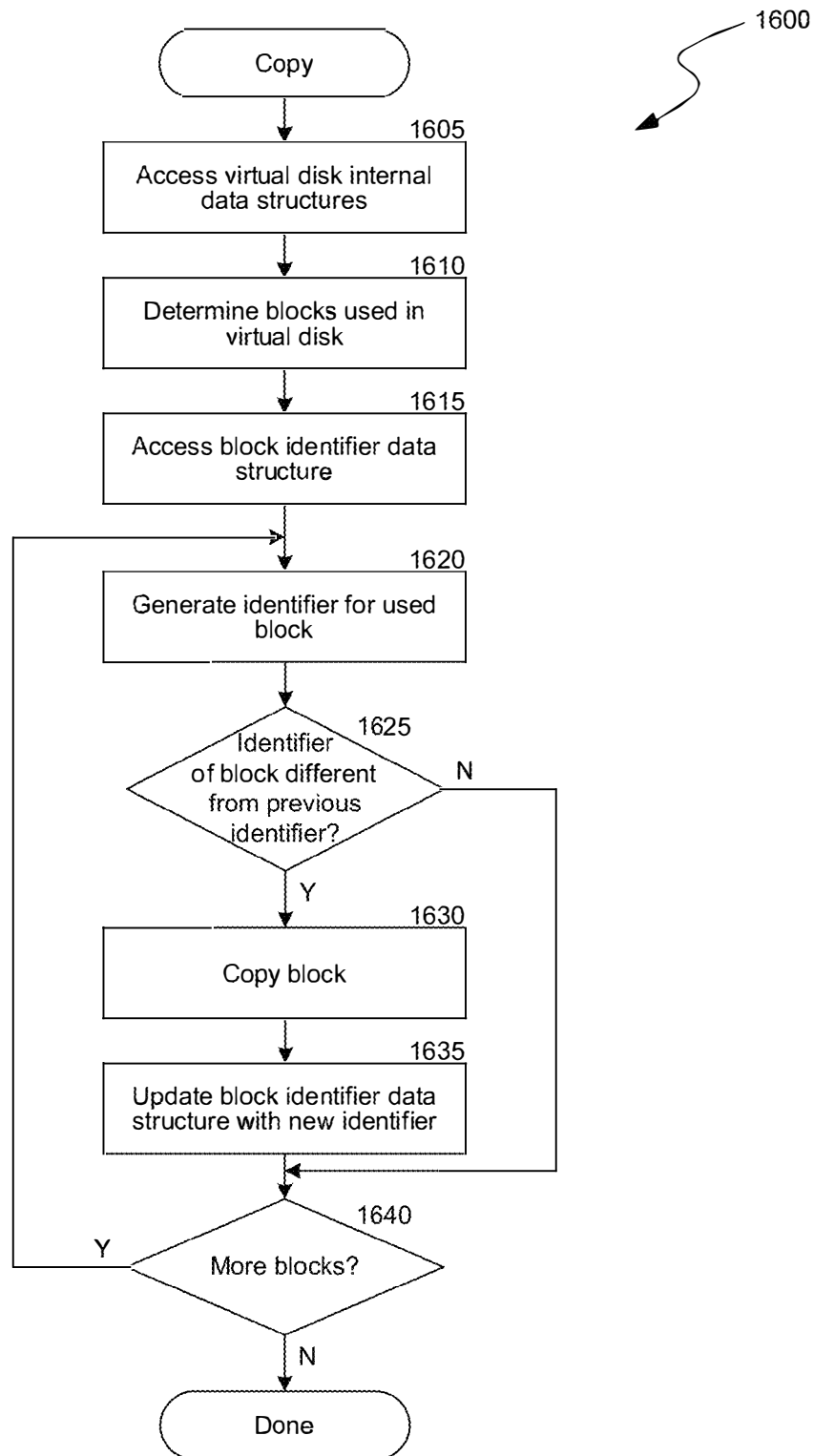


FIG. 15

U.S. Patent**Aug. 22, 2017****Sheet 20 of 21****US 9,740,723 B2****FIG. 16**

		1700
	Block Identifier	Substantially Unique Identifier
{	490	OxA1B3FG
	491	OxFG329A
	492	OxC1D839
	...	
	1702	1704

FIG. 17

US 9,740,723 B2

1

SYSTEMS AND METHODS FOR MANAGEMENT OF VIRTUALIZATION DATA

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a divisional of U.S. patent application Ser. No. 13/667,890 filed Nov. 2, 2012 (entitled SYSTEMS AND METHODS FOR MANAGEMENT OF VIRTUALIZATION DATA), now U.S. Pat. No. 8,725,973, which is a divisional of U.S. patent application Ser. No. 12/553,294 filed Sep. 3, 2009 (entitled SYSTEMS AND METHODS FOR MANAGEMENT OF VIRTUALIZATION DATA), now U.S. Pat. No. 8,307,177, which claims priority to U.S. Provisional Patent Application No. 61/094,753 filed Sep. 5, 2008 (entitled SYSTEMS AND METHODS FOR MANAGEMENT OF VIRTUALIZATION DATA), U.S. Provisional Patent Application No. 61/121,383 filed Dec. 10, 2008 (entitled SYSTEMS AND METHODS FOR MANAGEMENT OF VIRTUALIZATION DATA) and U.S. Provisional Patent Application No. 61/169,515 filed Apr. 15, 2009 (entitled SYSTEMS AND METHODS FOR MANAGEMENT OF VIRTUALIZATION DATA), each of which is incorporated by reference herein in its entirety.

BACKGROUND

In general, virtualization refers to the simultaneous hosting of one or more operating systems on a physical computer. Such virtual operating systems and their associated virtual resources are called virtual machines. Virtualization software sits between the virtual machines and the hardware of the physical computer. One example of virtualization software is ESX Server, by VMware, Inc. of Palo Alto, Calif. Other examples include Microsoft Virtual Server and Microsoft Windows Server Hyper-V, both by Microsoft Corporation of Redmond, Wash., and Sun xVM by Sun Microsystems Inc. of Santa Clara, Calif.

Virtualization software provides to each virtual operating system virtual resources, such as a virtual processor, virtual memory, a virtual network device, and a virtual disk. Each virtual machine has one or more virtual disks. Virtualization software typically stores the data of virtual disks in files on the filesystem of the physical computer, called virtual machine disk files (in the case of VMware virtual servers) or virtual hard disk image files (in the case of Microsoft virtual servers). For example, VMware's ESX Server provides the Virtual Machine File System (VMFS) for the storage of virtual machine disk files. A virtual machine reads data from and writes data to its virtual disk much the same way that an actual physical machine reads data from and writes data to an actual disk.

Traditionally, virtualization software vendors have enabled the backup of virtual machine data in one of two ways. A first method requires the installation of backup software on each virtual machine having data to be backed up and typically uses the same methods used to back up the data of physical computers to back up the virtual machine data. A second method backs up the files that store the virtual disks of the virtual machines, and may or may not require the installation of backup software on each virtual machine for which the data is to be backed up.

As an example of the second method, VMware Consolidated Backup (VCB), also by VMware, Inc., enables the backup of the data of virtual machines on ESX Server without having to install backup software on the virtual

2

machines. VCB consists of a set of utilities and scripts that work in conjunction with third-party backup software to backup virtual machine data. VCB and the third-party backup software are typically installed on a backup proxy server that uses the Microsoft Windows Server 2003 operating system by Microsoft Corporation. VCB supports file-level backups (backups at the level of files and directories) for virtual machines using Microsoft Windows operating systems. In a file-level backup, the granularity of the backup is at the level of individual files and/or directories of the virtual machine. A file-level backup allows copies of individual files on virtual disks to be made. File-level backups can be full backups, differential backups, or incremental backups.

VCB also supports image-level backups for virtual machines using any operating system (e.g., Microsoft Windows operating systems, Linux operating systems, or other operating systems that may be installed upon ESX Server). In an image-level backup, the granularity of the backup is at the level of a virtual machine (i.e., the entire virtual machine, including its current state is backed up). For an image-level backup, typically the virtual machine is suspended and all virtual disk and configuration files associated with the virtual machine are backed up, and then the virtual machine is resumed.

An administrator would typically choose to perform a file-level backup of a Microsoft Windows virtual machine because of the potential need to restore individual files or directories from the backed-up Microsoft virtual machine. However, VCB may not perform a file-level backup of a Microsoft Windows virtual machine as quickly as an image-level backup. Accordingly, a system that enables a backup of a Microsoft Windows virtual machine to be performed at least as quickly as a file-level backup and enables granular restoration of any data (e.g., individual files or directories) from the backed-up Microsoft virtual machine would have significant utility.

Because VCB only supports file-level backups for virtual machines using Microsoft Windows operating systems, a file-level backup cannot be performed using VCB for virtual machines using operating systems other than Microsoft Windows (e.g., Linux operating systems). An administrator must back up a non-Microsoft Windows virtual machine using an image-level backup. Therefore, in order to granularly restore data (e.g., an individual file or directory) from the backed-up non-Microsoft Windows virtual machine, the entire non-Microsoft Windows virtual machine must be restored. This may require overwriting the original virtual machine with the backed-up virtual machine, or re-creating the original virtual machine on a different physical machine. This may be a laborious and time-intensive process, and may result in loss of virtual machine data. Accordingly, a system that enables the granular restoration of any data (e.g., individual files or directories) within a virtual machine using any type of operating system would have significant utility.

Another challenge posed by the use of VCB to perform backups of virtual machines is that such backups require an administrator to manually identify or specify the virtual machines that are to be backed up, typically via a script created in advance of the backup operation. However, because virtual machines may be easily set up and torn down, virtual machines may be less permanent in nature than actual physical machines. Due to this potential transience of virtual machines, it may be more difficult for the administrator to identify all of the virtual machines which are to be backed up in advance of the backup operation. Accordingly, a system that provides automatic identification

US 9,740,723 B2

3

of virtual machines that are to be backed up at the time of the backup operation would have significant utility.

The need exists for a system that overcomes the above problems, as well as one that provides additional benefits. Overall, the examples herein of some prior or related systems and their associated limitations are intended to be illustrative and not exclusive. Other limitations of existing or prior systems will become apparent to those of skill in the art upon reading the following Detailed Description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGS. 1A and 1B are block diagrams illustrating environments in which aspects of the invention may be configured to operate.

FIG. 2 is a block diagram illustrating another environment in which aspects of the invention may be configured to operate.

FIG. 3 is a flow diagram illustrating a process for discovering one or more virtual machines.

FIGS. 4-6 are display diagrams illustrating example interfaces provided by aspects of the invention.

FIG. 7 is a flow diagram illustrating a process for copying virtual machine data.

FIG. 8 is a flow diagram illustrating a process for extracting metadata from virtual volumes and/or virtual disk and configuration files.

FIG. 9 is a flow diagram illustrating a process for restoring virtual machine data.

FIG. 10 is a block diagram illustrating an example of a data storage enterprise that may employ aspects of the invention.

FIGS. 11-15 are display diagrams illustrating example interfaces provided by aspects of the invention.

FIG. 16 is a flow diagram illustrating a process for copying virtual machine data.

FIG. 17 is a diagram illustrating a suitable data structure that may be employed by aspects of the invention.

DETAILED DESCRIPTION

The headings provided herein are for convenience only and do not necessarily affect the scope or meaning of the claimed invention.

Overview

Described in detail herein is a method of copying data of one or more virtual machines being hosted by one or more non-virtual machines. The method includes receiving an indication that specifies how to perform a copy of data of one or more virtual machines hosted by one or more virtual machine hosts. The method further includes determining whether the one or more virtual machines are managed by a virtual machine manager that manages or facilitates management of the virtual machines. If so, the virtual machine manager is dynamically queried to automatically determine the virtual machines that it manages or that it facilitates management of. If not, a virtual machine host is dynamically queried to automatically determine the virtual machines that it hosts. The data of each virtual machine is then copied according to the specifications of the received indication.

Under one example of the method, a file-level, volume-level or disk-level copy of a virtual machine is performed. Performing a file-level copy involves determining volumes of the virtual machine, mounting the volumes on a proxy server, and copying files from the volumes mounted on the proxy server to a secondary storage data store. Performing a volume-level copy involves determining volumes of the

4

virtual machine, mounting the volumes on a proxy server, and copying the volumes mounted on the proxy server to the secondary storage data store. Performing a disk-level copy involves determining virtual disk and configuration files of the virtual machine, copying the virtual disk and configuration files to the proxy server, extracting metadata from the virtual disk and configuration files, and copying the virtual disk and configuration files and the extracted metadata to the secondary storage data store.

Various examples of aspects of the invention will now be described. The following description provides specific details for a thorough understanding and enabling description of these examples. One skilled in the relevant art will understand, however, that aspects of the invention may be practiced without many of these details. Likewise, one skilled in the relevant art will also understand that aspects of the invention may have many other obvious features not described in detail herein. Additionally, some well-known structures or functions may not be shown or described in detail below, so as to avoid unnecessarily obscuring the relevant description.

The terminology used below is to be interpreted in its broadest reasonable manner, even though it is being used in conjunction with a detailed description of certain specific examples of aspects of the invention. Indeed, certain terms may even be emphasized below; however, any terminology intended to be interpreted in any restricted manner will be overtly and specifically defined as such in this Detailed Description section.

Unless described otherwise below, aspects of the invention may be practiced with conventional data processing systems. Thus, the construction and operation of the various blocks shown in FIGS. 1A, 1B and 2 may be of conventional design, and need not be described in further detail herein to make and use aspects of the invention, because such blocks will be understood by those skilled in the relevant art. One skilled in the relevant art can readily make any modifications necessary to the blocks in FIGS. 1A, 1B and 2 (or other embodiments or figures) based on the detailed description provided herein.

Aspects of the invention will now be described in detail with respect to FIGS. 1 through 17. FIGS. 1A, 1B and 2 are block diagrams illustrating various environments in which aspects of the invention may be configured to operate. FIG. 1A illustrates aspects of the invention interacting with virtual machines (e.g., VMware virtual machines or Microsoft virtual machines) storing data on a storage device connected to the virtual machine via a Storage Area Network (SAN), and FIG. 1B illustrates aspects of the invention interacting with virtual machines storing data locally. FIG. 2 illustrates aspects of the invention interacting with a virtual machine manager (e.g., a VMware Virtual Center server or a Microsoft System Center Virtual Machine Manager), which manages virtual machines. FIG. 3 is a flow diagram illustrating a process for discovering one or more virtual machines in one or more of the environments illustrated in FIGS. 1A, 1B and 2 (or in other environments).

FIGS. 4-6 are display diagrams illustrating example interfaces provided by aspects of the invention. An administrator (or other user) may use the example interfaces to administer storage operations, such as the copying of data of virtual machines. FIG. 7 is a flow diagram illustrating a process used by aspects of the invention to copy data of a virtual machine. FIG. 8 is a flow diagram illustrating a process for extracting metadata from virtual volumes and/or virtual disk and configuration files. FIG. 9 is a flow diagram illustrating a process for restoring virtual machine data. FIG. 10 is a

US 9,740,723 B2

5

block diagram illustrating an example of a data storage enterprise in which aspects of the invention may be configured to operate.

FIGS. 11, 12, 13A, 13B, 14A, and 14B are display diagrams illustrating example interfaces provided by aspects of the invention. The administrator may also use these example interfaces to administer storage operations, such as the restoration of data previously copied from virtual machines. FIG. 3 is a flow diagram illustrating a process that may be used in a storage operation to perform incremental copies of blocks of virtual machine data. FIG. 17 is a diagram illustrating a suitable data structure that may be used during the process of FIG. 16.

Suitable Environments

FIGS. 1A, 1B and 2 and the discussion herein provide a brief, general description of certain exemplary suitable computing environments in which aspects of the invention can be implemented. Although not required, aspects of the invention are described in the general context of computer-executable instructions, such as routines executed by a general-purpose computer, e.g., a server computer, wireless device, or personal computer. Those skilled in the relevant art will appreciate that aspects of the invention can be practiced with other communications, data processing, or computer system configurations, including: Internet appliances, hand-held devices (including personal digital assistants (PDAs)), wearable computers, all manner of wireless devices, multi-processor systems, microprocessor-based or programmable consumer electronics, set-top boxes, network PCs, mini-computers, mainframe computers, and the like. Indeed, the terms “computer,” “host,” and “host computer” are generally used interchangeably herein, and refer to any of the above or similar devices and systems, as well as any data processor.

Aspects of the invention can be embodied in a special purpose computer or data processor that is specifically programmed, configured, or constructed to perform one or more of the computer-executable instructions explained in detail herein. Aspects of the invention can also be practiced in distributed computing environments where tasks or modules are performed by remote processing devices, which are linked through a communications network, such as a Local Area Network (LAN), a Wide Area Network (WAN), a SAN, a Fibre Channel network, or the Internet. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Aspects of the invention may be stored or distributed on tangible computer-readable media, including magnetically or optically readable computer discs, hard-wired or preprogrammed chips (e.g., EEPROM semiconductor chips), nanotechnology memory, biological memory, or other tangible or physical data storage media. In some aspects of the system, computer implemented instructions, data structures, screen displays, and other data under aspects of the invention may be distributed over the Internet or over other networks (including wireless networks), on a propagated signal on a propagation medium (e.g., an electromagnetic wave(s), a sound wave, etc.) over a period of time, or they may be provided on any analog or digital network (packet switched, circuit switched, or other scheme). Those skilled in the relevant art will recognize that portions of aspects of the invention may reside on a server computer, while corresponding portions reside on a client computer such as a mobile or portable device, and thus, while certain hardware platforms are described herein, aspects of the invention are equally applicable to nodes on a network.

6

FIG. 1A is a block diagram illustrating an environment 100 in which aspects of the invention may be configured to operate. The environment 100 includes a virtual machine host 105 operating on or being hosted by a computing device 125, which may be a server. The environment 100 also includes a primary storage data store 135 connected to the computing device 125 via a SAN 130. The environment 100 also includes a virtual machine storage manager 145 operating on or being hosted by another computing device 170, which may be another server, and a secondary storage data store 175 connected to the computing device 170. The computing devices 125 and 170 are connected to each other via a network 180, which may be a LAN, a WAN, the public Internet, some other type of network, or some combination of the above.

The virtual machine host 105 hosts one or more virtual machines 110 (shown individually as virtual machines 110a and 110b). Each virtual machine 110 has its own operating system 120 (shown individually as operating systems 120a and 120b) and one or more applications 115 executing on the operating system or loaded on the operating system (shown individually as applications 115a and 115b). The operating systems 120 may be any type of operating system 120 (e.g., Microsoft Windows 95/98/NT/2000/XP/2003/2008, Linux operating systems, Sun Solaris operating systems, UNIX operating systems, etc.) that can be hosted by the virtual machine host 105. The applications 115 may be any applications (e.g., database applications, file server applications, mail server applications, web server applications, transaction processing applications, etc.) that may run on the operating systems 120. The virtual machines 110 are also connected to the network 180.

The computing device 125 is connected to the primary storage data store 135 via the SAN 130, which may be any type of SAN (e.g., a Fibre Channel SAN, an iSCSI SAN, or any other type of SAN). The primary storage data store 135 stores the virtual disks 140 (shown individually as virtual disks 140a and 140b) of the virtual machines 110 hosted by the virtual machine host 105. Virtual disk 140a is used by virtual machine 110a, and virtual disk 140b is used by virtual machine 110b. Although each virtual machine 110 is shown with only one virtual disk 140, each virtual machine 110 may have more than one virtual disk 140 in the primary storage data store 135. As described in more detail herein, a virtual disk 140 corresponds to one or more files (e.g., one or more *.vmdk or *.vhd files) on the primary storage data store 135. The primary storage data store 135 stores a primary copy of the data of the virtual machines 110. Additionally or alternatively, the virtual disks 140 may be stored by other storage devices in the environment 100.

A primary copy of data generally includes a production copy or other “live” version of the data that is used by a software application and is generally in the native format of that application. Primary copy data may be maintained in a local memory or other high-speed storage device (e.g., on the virtual disks 140 located in the primary storage data store 135) that allows for relatively fast data access if necessary. Such primary copy data may be intended for short-term retention (e.g., several hours or days) before some or all of the data is stored as one or more secondary copies, for example, to prevent loss of data in the event a problem occurs with the data stored in primary storage.

In contrast, secondary copies include point-in-time data and are typically intended for long-term retention (e.g., weeks, months, or years depending on retention criteria, for example, as specified in a storage or retention policy) before some or all of the data is moved to other storage or

US 9,740,723 B2

7

discarded. Secondary copies may be indexed so users can browse and restore the data at another point in time. After certain primary copy data is backed up, a pointer or other location indicia, such as a stub, may be placed in the primary copy to indicate the current location of that data. The secondary storage data store **175** stores one or more secondary copies of the data of the virtual machines **110**.

The virtual machine storage manager **145** includes a virtual machine storage operation component **150**, which includes a Virtual Logical Unit Number (VLUN) driver **152** (for accessing virtual disks **140**, described in more detail herein) and a virtual machine mount component **154** (for mounting virtual machines, described in more detail herein). The virtual machine storage manager **145** also includes a data agent **155**. The data agent **155** includes an integration component **157** that provides functionality for the virtual machine storage operation component **150**. The data agent **155** also includes a virtual disk analyzer component **160** that examines the virtual disk and configuration files corresponding to the virtual disks **140** and extracts metadata from the virtual disk and configuration files. For example, the integration component **157** may include a set of scripts that the data agent **155** causes to be run prior to, during, and/or following a copy of virtual machine data. As another example, the integration component **157** may be a component that encapsulates or wraps the virtual machine mount component **154** and provides an Application Programming Interface (API) with functions for accessing the virtual machine mount component **154**. The virtual machine storage manager **145** also includes a data store **165** that maintains data used by the virtual machine storage manager **145**, such as data used during storage operations, and configuration data.

The secondary storage data store **175** is connected to the computing device **170**. The secondary storage data store **175** may be any type of storage suitable for storing one or more secondary copies of data, such as Directly-Attached Storage (DAS) such as hard disks, storage devices connected via another SAN (e.g., a Fibre Channel SAN, an iSCSI SAN, or any other type of SAN), Network-Attached Storage (NAS), a tape library, optical storage, or any other type of storage. The secondary storage data store **175** stores virtual machine data that is copied by the virtual machine storage manager **145**. Accordingly, the secondary storage data store **175** stores one or more secondary copies, of the data of the virtual machines **110**. A secondary copy can be in one or more various formats (e.g., a copy set, a backup set, an archival set, a migration set, etc.).

FIG. 1B is a block diagram illustrating another environment **101** in which aspects of the invention may be configured to operate. The environment **101** is substantially the same as the environment **100** illustrated in FIG. 1A, except that primary storage data store **135** resides in the computing device **125** hosting the virtual machine host **105** (the primary storage data store **135** is local storage). The local primary storage data store **135** includes a virtual disk **140a** for use by virtual machine **110a**, and a virtual disk **140b** for use by virtual machine **110b**. In addition to or as an alternative to the primary storage data stores **135** illustrated in FIGS. 1A and 1B, the virtual machine host **105** may use other methods of storing data, such as Raw Device Mapping (RDM) on a local or network-attached device (NAS) or on storage devices connected via another SAN.

FIG. 2 is a block diagram illustrating yet another environment **200** in which aspects of the invention may be configured to operate. The environment **200** includes two computing devices **125** (shown individually as computing

8

devices **125a** and **125b**), each hosting a virtual machine host **105** (shown individually as virtual machine hosts **105a** and **105b**). The primary storage data store **135** includes two additional virtual disks **140c** and **140d** that store the data of virtual machines **110c** and **110d**, respectively.

The environment **200** also includes a virtual machine manager **202** operating on a computing device **215** (e.g., a server). The virtual machine manager **202** includes a virtual machine management component **205** which enables administrators (or other users with the appropriate permissions; the term administrator is used herein for brevity) to manage the virtual machines **110**. The virtual machine manager **202** also includes an Application Programming Interface (API) component **210**, which provides functions that enable the data agent **155** to programmatically interact with the virtual machine manager **202** and the virtual machines **110**. The virtual machine hosts **105** may also each include an API component. The virtual machine manager **202** and/or the virtual machine hosts **105** may expose or provide other APIs not illustrated in FIG. 1A, 1B or 2, such as an API for accessing and manipulating virtual disks **140**, and APIs for performing other functions related to management of virtual machines **110**.

The environments **100**, **101** and **200** may include components other than those illustrated in FIGS. 1A, 1B and 2, respectively, and the components illustrated may perform functions other than or in addition to those described herein. For example, the virtual machine storage manager **145** may include a public key certificate (e.g., an X.509 public key certificate) that the virtual machine storage manager **145** provides to the virtual machine host **105** or the virtual machine manager **202**. The virtual machine host **105** or the virtual machine manager **202** can then use the X.509 public key of the certificate to encrypt data that is to be transmitted to the virtual machine storage manager **145**. As another example, the network **180** may include a firewall that sits between the virtual machine host **105** and the virtual machine storage manager **145**, and data being copied may have to pass through the firewall. If this is the case, the virtual machine storage manager **145** may use the systems and methods described in commonly-assigned U.S. patent application Ser. No. 10/818,747 (entitled SYSTEM AND METHOD FOR PERFORMING STORAGE OPERATIONS THROUGH A FIREWALL), the entirety of which is incorporated by reference herein.

As another example, a secondary storage computing device (which is described in more detail herein, e.g., with reference to FIG. 10) may be connected to the virtual machine storage manager **145** and to the secondary storage data store **175**. The secondary storage computing device may assist in the transfer of copy data from the virtual machine storage manager **145** to the secondary storage data store **175**. The secondary storage computing device may perform functions such as encrypting, compressing, single or variable instancing, and/or indexing data that is transferred to the secondary storage data store **175**. As another example, one or more agents (e.g., a file system agent and/or a proxy host agent) as well as a set of utilities (e.g., VMware Tools if the virtual machines **110** are VMware virtual machines) may reside on each virtual machine **110** to provide functionality associated with copying and restoring virtual machine data. As another example, the environments **100**, **101** and **200** may include components or agents that perform various functions on virtual machine and other data, such as classifying data, indexing data, and single or vari-

US 9,740,723 B2

9

able instanting or deduplicating data at different phases of storage operations performed on virtual machine and other data.

As another example, the secondary storage data store **175** may include one or more single instance storage devices that store only a single instance of multiple instances of data (e.g., only a single instance of multiple instances of identical files or data objects stored on one or more computing devices). If this is the case, the secondary storage data store **175** may include one or more single instance storage devices as described in one or more of the following commonly-assigned U.S. patent applications: 1) U.S. patent application Ser. No. 11/269,512 (entitled SYSTEM AND METHOD TO SUPPORT SINGLE INSTANCE STORAGE OPERATIONS); 2) U.S. patent application Ser. No. 12/145,347 (entitled APPLICATION-AWARE AND REMOTE SINGLE INSTANCE DATA MANAGEMENT); or 3) U.S. patent application Ser. No. 12/145,342 (entitled APPLICATION-AWARE AND REMOTE SINGLE INSTANCE DATA MANAGEMENT); 4) U.S. patent application Ser. No. 11/963,623 (entitled SYSTEM AND METHOD FOR STORING REDUNDANT INFORMATION); 5) U.S. patent application Ser. No. 11/950,376 (entitled SYSTEMS AND METHODS FOR CREATING COPIES OF DATA SUCH AS ARCHIVE COPIES); or 6) U.S. Pat. App. No. 61/100,686 (entitled SYSTEMS AND METHODS FOR MANAGING SINGLE INSTANCING DATA), each of which is incorporated by reference herein in its entirety.

As a further example, the secondary storage data store **175** may include one or more variable instance storage devices that store a variable number of instances of data (e.g., a variable number of instances of identical files or data objects stored on one or more computing devices). If this is the case, the secondary storage data store **175** may include one or more variable instance storage devices as described in the following commonly-assigned U.S. Pat. App. No. 61/164,803 (entitled STORING A VARIABLE NUMBER OF INSTANCES OF DATA OBJECTS).

Example Layouts of Virtual Disks

Virtual disks **140**, as used in the systems described in FIGS. 1A, 1B and 2, may have various configurations. As previously described, a virtual disk **140** corresponds to one or more virtual disk files (e.g., one or more *.vmdk or *.vhd files) on the primary storage datastore **135**. A virtual machine host **105** may support several types of virtual disks **140**. For example, a virtual disk **140** may be either: 1) a growable virtual disk **140** contained in a single virtual disk file that can grow in size (e.g., a monolithic sparse virtual disk that starts at 2 GB and grows larger); 2) a growable virtual disk **140** split into multiple virtual disk files (e.g., a split sparse virtual disk comprising multiple 2 GB virtual disk files), the aggregation of which can grow in size by adding new virtual disk files; 3) a preallocated virtual disk **140** contained in a single virtual disk file (e.g., a monolithic flat virtual disk, the size of which does not change); or 4) a preallocated virtual disk **140** split into multiple virtual disk files (e.g., a split flat virtual disk comprising multiple 2 GB virtual disk files, the number of which and the size of each of which does not change). Where a virtual disk **140** is split into multiple virtual disk files, each individual virtual disk file is called an extent. A virtual machine host **105** may also support types of virtual disks **140** other than these types. Those of skill in the art will understand that a virtual disk **140** can be structured in a wide variety of configurations, and that virtual disks **140** are not limited to the configurations described herein.

10

A virtual machine host **105** may support snapshotting, or taking a snapshot of a virtual machine **110**. The virtual machine host **105** can snapshot a virtual machine **110** in a linear fashion (in which there is only one branch of snapshots from the original state of the virtual machine **110**, and each snapshot in the branch linearly progresses from prior snapshots) or in a process tree (in which there are multiple branches of snapshots from the original state of the virtual machine **110**, and two snapshots may or may not be in the same branch from the original state of the virtual machine **110**). When a snapshot is taken of a virtual machine **110**, the virtual machine **110** stops writing to its virtual disks **140** (e.g., stops writing to the one or more *.vmdk files). The virtual machine **110** writes future writes to a delta disk file (e.g., a *.delta.vmdk file) using, for example, a copy-on-write (COW) semantic. As the virtual machine host **105** can snapshot a virtual machine **110** repeatedly, there can be multiple delta disk files. The virtual disk and delta disk files can be analogized to links in a chain. Using this analogy, the original disk file is a first link in the chain. A first child delta disk file is a second link in the chain, and a second child delta disk file is a third link in the chain, and so forth.

Also as previously described, a virtual machine **110** generally has associated configuration files that a virtual machine host **105** uses to store configuration data about the virtual machine **110**. These configuration files may include a *.vmx file, which stores data about the parent-child relationships created between virtual disk files and delta disk files when a snapshot of a virtual machine **110** is taken. These configuration files may also include a disk descriptor file (e.g., a *.vmdk file). In some embodiments, instead of using a disk descriptor file, the disk descriptor is embedded into a virtual disk file (e.g., embedded in a *.vmdk file).

The disk descriptor file generally stores data about the virtual disk files that make up a virtual disk **140**. This data includes information about the type of the virtual disk **140**. For example, the virtual disk **140** may be a monolithic flat virtual disk, a monolithic sparse virtual disk, a split flat virtual disk, a split sparse virtual disk or another type of a virtual disk. This data also includes an identifier of the parent of the virtual disk file, if it has one (if the virtual machine **110** has been snapshotted, its original virtual disk file will have a child virtual disk file), a disk database describing geometry values for the virtual disk **140** (e.g., cylinders, heads and sectors) and information describing the extents that make up the virtual disk **140**. Each extent may be described by a line in the disk descriptor file having the following format:

[type of access] [size] [type] [file name of extent]

Following is an example of a line in the disk descriptor file describing an extent:

RW 16777216 VMFS "test-flat.vmdk"

This line describes an extent for which read/write access is allowed, of size 16777216 sectors, of type VMFS (e.g., for use on a primary storage data store **135**), and the filename of the virtual disk file—"test-flat.vmdk."

A virtual machine host **105** provides an abstraction layer such that the one or more virtual disks files (and any delta disk files) of the virtual disks **140** appear as one or more actual disks (e.g., one or more hard disk drives) to a virtual machine **110**. Because the virtual machine host **105** abstracts the virtual disk **140** so that it appears as an actual disk to an operating system **120** executing on the virtual machine **110**, the operating system **120** can generally use its standard file system for storing data on a virtual disk **140**. The various structures used by the file system and the operating system **120** (e.g., the partition table(s), the volume manager data-

US 9,740,723 B2

11

base(s) and the file allocation table(s)) are stored in the one or more virtual disk files that make up a virtual disk **140**.

For example, a virtual machine host **105** may store a single virtual disk file (e.g., a single *.vmdk file) that is a preallocated virtual disk **140** (a monolithic flat virtual disk) for each virtual disk used by a virtual machine **110** operating on the virtual machine host **105**. The single virtual disk file may be named <virtual machine name>-flat.vmdk. There would also be a disk descriptor file for the single virtual disk file that would typically be named <virtual machine name>-.vmdk. A snapshot taken of the virtual machine **110** would result in an additional delta disk file being created that is a single virtual disk file (e.g., a single *.vmdk file), which is a growable virtual disk **140** (a monolithic sparse virtual disk). The delta disk file would typically be named <virtual disk name>-<#####>-delta.vmdk, where <#####> is a number indicating the sequence of the snapshot. There would also be a disk descriptor file for the single virtual disk file that would typically be named <virtual disk name>-<#####>-.vmdk, again, where <#####> is a number indicating the sequence of the snapshot.

Process for Discovering Virtual Machines

FIG. 3 is a flow diagram illustrating a process for discovering one or more virtual machines **110** (e.g., for an operation to copy their data). In general, for ease in describing features of the invention, aspects of the invention will now be described in terms of a user (e.g., an administrator) interacting with the server computer via his or her user computer. As implemented, however, the user computer receives data input by the user and transmits such input data to the server computer. The server computer then queries the database, retrieves requested pages, performs computations and/or provides output data back to the user computer, typically for visual display to the user. Thus, for example, under step **305**, a user provides input specifying that a copy operation is to be performed and how to perform the copy operation. The data agent **155** receives this input and performs the copy operation according to the input.

The process **300** begins at step **305** when the data agent **155** receives an indication specifying that the data agent **155** is to perform a copy operation and how to perform the copy operation. The indication may be received from the administrator (e.g., a manually-specified indication to perform a copy operation) or be triggered automatically (e.g., by an automated schedule). The indication may be received as a result of a storage policy that specifies how and/or when to copy data from one or more virtual machines **110** to the secondary storage data store **175**.

A storage policy is generally a data structure or other information source that includes a set of preferences and other storage criteria associated with performing a storage operation. The preferences and storage criteria may include, but are not limited to, a storage location, relationships between system components, network pathways to utilize in a storage operation, retention policies, data characteristics, compression or encryption requirements, preferred system components to utilize in a storage operation, a single-instancing or variable instancing policy to apply to the data, and other criteria relating to a storage operation. For example, a storage policy may indicate that certain data is to be stored in the secondary storage data store **175**, retained for a specified period of time before being aged to another tier of secondary storage, copied to the secondary storage data store **175** using a specified number of data streams, etc. A storage policy may be stored in a database of a storage manager (see, e.g., FIG. 10 and accompanying description), to archive media as metadata for use in restore operations or

12

other storage operations, or to other locations or components of the system. The storage manager may include a jobs agent that monitors the status of some or all storage operations previously performed, currently being performed, or scheduled to be performed.

For example, an administrator may create a storage policy for copying data of virtual machines **110** and perform the copying of their data to the secondary storage data store **175** according to the storage policy. This storage policy may specify that the virtual machine storage manager **145** is to perform a file-level copy of certain files (e.g., all files in a specific directory or satisfying selection criteria) on multiple virtual machines **110**. As yet another example, the storage policy may specify that the virtual machine storage manager **145** is to perform a volume-level copy of all virtual machines **110** on multiple virtual machine hosts **105**. As another example, the storage policy may specify that the virtual machine storage manager **145** is to perform a disk-level copy of all virtual machines **110** on all virtual machine hosts **105** associated with a virtual machine manager **202**. File-level, volume-level and disk-level copying is discussed in more detail herein, for example, with reference to FIG. 7.

At decision step **310** the data agent **155** determines (e.g., by reading a stored indication of the virtual machine manager **202**, or by scanning a network for a virtual machine manager **202**) whether there is a virtual machine manager **202** managing the virtual machine hosts **105** and associated virtual machines **110**. If there is a virtual machine manager **202**, the process **300** continues at step **325**, where the data agent **155** queries the virtual machine manager **202** to determine the virtual machines **110** that it manages and to receive an ordered or unordered list of virtual machines **110**. The data agent **155** may call a function of the API component **210** to determine the virtual machines **110** managed by the virtual machine manager **202** and receive an ordered or unordered list of virtual machines **110**.

If there is not a virtual machine manager **202**, the process **300** continues at step **315**, where the data agent **155** selects the next virtual machine host **105**, which, on the first loop, is the first determined virtual machine host **105**. The virtual machine hosts **105** may be dynamically determined (e.g., by scanning the network **180**) or determined statically (e.g., by reading a stored indication of the virtual machine hosts **105**). More details as to the detection of virtual machine hosts **105** and virtual machines **110** are described herein for example, with reference to FIGS. 4-6. The steps **325** and **320** are not to be understood as mutually exclusive. For example, the data agent **155** may determine a first set of virtual machines **110** by accessing the virtual machine manager **202**, and a second set of virtual machines by accessing one or more virtual machine hosts **105**.

At step **320** the data agent **155** queries the virtual machine host **105** to determine the virtual machines **110** that it hosts. The data agent **155** may call a function of the API component **210** to determine the virtual machines **110** hosted by the virtual machine host **105** and to receive an ordered or unordered list of virtual machines **110**. At step **330**, the data agent **155** begins looping through the list of virtual machines **110** that it determined in either or both of steps **320** or **325** and selects the next virtual machine **110** on the list, which, on the first loop, is the first determined virtual machine **110**. At step **335** the data agent **155** copies the data of the virtual machine, for example, according to the indication received in step **305**, or according to a storage policy. This process is described in more detail herein, for example, with reference to FIG. 7.

US 9,740,723 B2

13

At step 337, other processing of virtual machine data may be performed. For example, the data agent 155 (or another agent, such as a data classification agent) may analyze and classify the virtual machine data. To do so, the data agent 155 may use techniques such as those described in commonly assigned U.S. patent application Ser. No. 11/564,119 (entitled SYSTEMS AND METHODS FOR CLASSIFYING AND TRANSFERRING INFORMATION IN A STORAGE NETWORK), the entirety of which is incorporated by reference herein. As another example, the data agent 155 (or another agent, such as an indexing agent) may create an index of the virtual machine data. To do so, the data agent 155 may use techniques such as those described in commonly-assigned U.S. patent application Ser. No. 11/694,869 (entitled METHOD AND SYSTEM FOR OFFLINE INDEXING OF CONTENT AND CLASSIFYING STORED DATA), the entirety of which is incorporated herein. As a final example, the data agent 155 may single or variable instance or de-duplicate the virtual machine data. To do so, the data agent 155 may use techniques described in one or more of previously-referenced U.S. patent application Ser. Nos. 11/269,512, 12/145,347, 12/145,342, 11/963,623, 11/950,376, 61/100,686, and 61/164,803. At decision step 340, the data agent 155 determines whether there are more virtual machines 110 for which the data is to be copied. If so, the data agent 155 returns to step 330, where the next virtual machine 110 is selected.

If there are no more virtual machines 110 for which the data is to be copied (e.g., if the data agent 155 has looped through the list of all the virtual machines 110 determined in either or both of steps 320 or 325), the process continues at step 345. At decision step 345, if there is not a virtual machine manager 202 (e.g., as determined in decision step 310), the data agent 155 determines whether there are more virtual machine hosts 105 (e.g., if more than one virtual machine hosts 105 was specified in the indication received in step 305). If there are more virtual machine hosts 105, the data agent 155 returns to step 315. If not, the process 300 concludes.

Interfaces for Configuring Storage Operations for Virtual Machine Data

Referring to FIGS. 4 through 6, representative computer displays or web pages for configuring storage operations to be performed for virtual machine data will now be described. The screens of FIGS. 4 through 6 may be implemented in any of various ways, such as in C++ or as web pages in XML (Extensible Markup Language), HTML (Hyper Text Markup Language), or any other scripts or methods of creating displayable data, such as the Wireless Access Protocol ("WAP"). The screens or web pages provide facilities to present information and receive input data, such as a form or page with fields to be filled in, pull-down menus or entries allowing one or more of several options to be selected, buttons, sliders, hypertext links or other known user interface tools for receiving user input. While certain ways of displaying information to users is shown and described with respect to certain Figures, those skilled in the relevant art will recognize that various other alternatives may be employed. The terms "screen," "web page" and "page" are generally used interchangeably herein.

When implemented as web pages, the screens are stored as display descriptions, graphical user interfaces, or other methods of depicting information on a computer screen (e.g., commands, links, fonts, colors, layout, sizes and relative positions, and the like), where the layout and information or content to be displayed on the page is stored in a database typically connected to a server. In general, a

14

"link" refers to any resource locator identifying a resource on a network, such as a display description provided by an organization having a site or node on the network. A "display description," as generally used herein, refers to any method of automatically displaying information on a computer screen in any of the above-noted formats, as well as other formats, such as email or character/code-based formats, algorithm-based formats (e.g., vector generated), or matrix or bit-mapped formats. While aspects of the invention are described herein using a networked environment, some or all features may be implemented within a single-computer environment.

FIG. 4 is a display diagram illustrating an example interface 400 provided by aspects of the invention. The interface 400 enables an administrator to specify options for the data agent 155 to discover virtual machines 110 for purposes of adding them to a sub-client. Clients and sub-clients are discussed in more detail with respect to FIGS. 5A and 5B. The administrator can specify that the data agent 155 is to automatically discover virtual machines 110 by selecting check box 405, which enables two options. The first option, which can be chosen by selecting radio button 410a and using the button 415a labeled "Configure," specifies that the data agent 155 is to discover virtual machine hosts 105 or virtual machines 110 that match a regular expression (e.g., an expression that describes a set of strings). The second option, which can be chosen by selecting radio button 410b and using the button 415b labeled "Configure," specifies that the data agent 155 is to discover virtual machines 110 associated with one or more specified virtual machine hosts 105. The two options allow the administrator to specify one or more criteria (e.g., based on names of the virtual machines 110) that discovered virtual machine 110 should meet in order to be associated with a storage policy, and this to have storage operations performed upon their data. Additionally or alternatively, data of virtual machines 110 can be classified or categorized (e.g., using techniques described in the previously referenced U.S. patent application Ser. No. 11/564,119) and the one or more criteria can use these classifications or categorizations. Detected virtual machines 110 that meet these one or more criteria (or having data that meets these one or more criteria) can be associated with a storage policy. This allows storage operations to be performed upon their data.

Buttons 418 enable the administrator to confirm or cancel the selections and/or view help regarding the interface 400. The interface 400 may enable discovery of virtual machines 110 by both regular expression matching and by association with one or more specified virtual machine hosts 105. For example, the interface 400 could be configured to discover all virtual machines 110 associated with a specific virtual machine host 105, as well as an additional number of virtual machines 110 having names that match a regular expression (e.g., "virtual/A").

FIG. 5A is a display diagram illustrating another example interface 500 provided by aspects of the invention. Tab 510 specifies general options that may be configurable by an administrator. Tab 510 also specifies a virtual machine storage manager 145, name 512, an application 514, an instance name 516, and a backup set name 518. The name 512 corresponds to the virtual machine storage manager 145 that hosts the data agent 155. The administrator can establish one or more sub-clients for the virtual machine storage manager 145. A sub-client is a portion of a client, and can contain either all of the client's data or a designated subset thereof. A default sub-client may be established for the data agent 155 that provides for protection of substantially all of

US 9,740,723 B2

15

the client's data (e.g., the data of the virtual machines 110). Protection of data generally refers to performing a storage operation on a primary copy of data to produce one or more secondary copies of the data. Storage operations performed to protect data may include copy operations, backup operations, snapshot operations, Hierarchical Storage Management (HSM) operations, migration operations, archive operations, and other types of storage operations known to those of skill in the art.

An administrator can also establish additional sub-clients to provide a further level of protection of virtual machine data. For example, for a virtual machine 110 upon which is loaded a mail application (e.g., a Microsoft Exchange mail server) and a database application (e.g., an Oracle database application), the administrator could establish one sub-client for protection of the data of the mail application (e.g., user mailboxes) and one sub-client for protection of the data of the database application (e.g., databases, datafiles and/or tablespaces). As another example, the administrator could establish sub-clients for organizational groupings (e.g., a sub-client for a marketing group, a sub-client for a sales group, etc.) and/or for virtual machines 110 based upon their purpose (e.g., a sub-client for virtual machines 110 used in production settings, a sub-client for virtual machines 110 used in test and/or development settings, etc.). Those of skill in the art will understand that an administrator may establish sub-clients according to various groupings.

An administrator can specify that any newly discovered virtual machines 110 that do not qualify for membership in an established sub-client group are to be added to the default sub-client by selecting check box 520. The administrator can also select that the data agent 155 is to discover virtual machines 110 and add them to particular sub-clients based upon rules by selecting check box 522. Since check box 522 is not selected, the options below it may not be selectable. However, selecting check box 522 allows the administrator to select radio buttons 524 (shown individually as radio buttons 524a and 524b) and buttons 526 (shown individually as buttons 526a and 526b), which enable functionality similar to that discussed with reference to the interface 400 of FIG. 4. For example, selection of the radio button 524a and the button 526a can enable the administrator to specify that all virtual machines 110 that match a regular expression (e.g., the regular expression "[a-g]" could be used to match any virtual machines 110 (or any virtual machine hosts 105) beginning with names for which the first character begins with any character in the range of "a" to "g"), are to be added to a particular sub-client. As another example, selection of the radio button 524b and the button 526b can enable the administrator to specify that all virtual machines 110 that are associated with a particular virtual machine host 105 (that is identified by e.g., name, IP address, and/or other identifier) are to be added to a particular sub-client. Buttons 528 enable the administrator to confirm or cancel the selections and/or view help regarding the interface 500.

FIG. 5B is a display diagram illustrating another example interface 550 provided by aspects of the invention that is shown when tab 530 is selected. Tab 530 specifies the configurations of virtual machine hosts 105, virtual machines 110, and sub-clients. The tab 530 displays three columns, column 542, labeled "Virtual machine host," column 544, labeled "Virtual machine," and column 546, labeled "Sub-client name." Column 542 contains a listing of discovered virtual machine hosts 105. Column 544 contains a listing of discovered virtual machines 110, grouped by their associated virtual machine hosts 105. Column 546 contains a listing of sub-client names that are associated

16

with each virtual machine 110. The virtual machine hosts 105 are divided into three groups 560, 562, and 564. Several of the virtual machines 110 hosted by the first group of virtual machines servers 105 have sub-client names of "Sub-client_test," indicating that they are part of this sub-client. The last virtual machine 110 in group 560 displays a list box 570 listing three different sub-clients that may be selected for the virtual machine 110 named "rack0102rh4x64." The virtual machine 110 is currently part of the sub-client "Sub-client_test," but other sub-clients may be selected. The virtual machines 110 that are part of the same sub-client have the same storage policy applied to protect their data.

Other virtual machines 110 that are part of the groups 562 or 564 are shown as being part of other sub-clients, such as the virtual machine 110 named "VM2" that is part of a sub-client 572 named "Database_SC," which may be a sub-client directed toward protecting data of a database application, and the virtual machine 110 named "VM3" that is part of a sub-client 574 named "Filesrv_SC," which may be a sub-client directed toward protecting data on a file server. Similarly, the virtual machines 110 named "SG111" and "SG3(1)" are both part of a sub-client 576 named "Marketing_Sales_SC," which may be a sub-client directed toward protecting data of marketing and sales organizations. The virtual machine 110 named "W2K8_SC" is also part of the sub-client 574. Accordingly, two different virtual machines 110 on two different virtual machine hosts 105 may be part of the same sub-client.

The sub-client 574 may also include other, non-virtual machines. (Non-virtual machines can be defined broadly to include operating systems on computing devices that are not virtualized. For example, the operating systems of the virtual machine hosts 105, the virtual machine manager 202, and the virtual machine storage manager 145 can be considered to be non-virtual machines.) In this case, the same storage policy would be applied to protect data of both the associated virtual machines 115 and the non-virtual machines. An administrator can select one or more virtual machine hosts 105 and select a sub-client using the listbox 554, and then select the button 556 labeled "Apply" to change all of the selected virtual machine hosts 105 to a selected sub-client. When the administrator selects the button 552 labeled "Discover," an automated process for discovering virtual machine hosts 105 and/or virtual machines 110 is started. When it concludes, the interface 550 displays any virtual machine hosts 105 and/or virtual machines 110 discovered by the process. Buttons 578 enable the administrator to confirm or cancel the selections and/or view help regarding the interface 550.

FIG. 6 is a display diagram illustrating another example interface 600 provided by aspects of the invention. The interface 600 enables the administrator to specify virtual machine hosts 105 and/or virtual machines 110 and security credentials for the data agent 155 to use when accessing the virtual machines 110. For example, the virtual machine hosts 105 and/or the virtual machines 110 may use well-known authentication and authorization technicians (e.g., username and password, and/or access control lists (ACLs)) to control access to virtual machine data. The interface 600 includes a region 610 in which a listing of virtual machine hosts 105 and/or virtual machines 110 can be shown. The administrator can add, edit and/or remove virtual machine hosts 105 and/or virtual machines 110 by selecting buttons 612, 614 and/or 616, respectively. Buttons 618 enable the administrator to confirm or cancel the selections and/or view help regarding the interface 600.

Process for Copying Virtual Machine Data

FIG. 7 is a flow diagram illustrating a process 700 for copying data of a virtual machine 110. (E.g., according to the indication received in step 305 of the process 300 of FIG. 3, or according to a storage policy.) One or more of the entities illustrated in the figures (e.g., FIGS. 1A, 1B, 2, and/or 10) may perform different aspects of the process 700. In some examples, a storage manager 1005 instigates the process 700 by sending an indication specifying the storage operation to the data agent 155 on the virtual machine storage manager 145. The data agent 155 performs the copying of the data of the virtual machine 110. The data agent 155 sends the data to a secondary storage computing device 1065, which then stores the data on one or more storage devices 1015 (e.g., the secondary storage data store 175). In some examples, less than all of these entities may be involved in performing the storage operation. The processes described herein are indicated as being performed by the data agent 155, although those of skill in the art will understand that aspects of the process 700 may be performed by any one of the entities described herein (e.g., the storage manager 1005, the secondary storage computing device 1065, etc.).

As previously described, the integration component 157 encapsulates the virtual machine mount component 154 and provides an API for accessing the virtual machine mount component 154. For example, if the virtual machines 110 are VMware virtual machines, the virtual machine mount component 154 may be VMware's vcbMounter command-line tool, and the integration component 157 may encapsulate the functionality provided by vcbMounter into an API and redirect the output of the vcbMounter tool. At step 705, the data agent 155 calls an API function of the integration component 157 to quiesce the file systems of the virtual machine 110. Quiescing the file systems ensures that no file system writes are pending at the time a snapshot of a virtual machine 110 is taken, thereby allowing the creation of filesystem-consistent copies. The data agent 155 may, prior to quiescing the file systems in step 705, also quiesce applications that are executing on the virtual machine 110 or are loaded on the virtual machine 110.

At step 710, the data agent 155 calls an API function of the integration component 157 to put the virtual machine 110 into snapshot mode. Alternatively, the data agent 155 may call a function of the API component 210 to put the virtual machine 110 into snapshot mode. When the virtual machine 110 is put into snapshot mode, the virtual machine 110 stops writing to its virtual disks 140 (e.g., stops writing to the one or more *.vmdk files or *.vhd files) on the primary storage data store 135. The virtual machine 110 writes future writes to a delta disk file (e.g., a *delta.vmdk file) on the primary storage data store 135. Putting the virtual machine 110 into snapshot mode enables the virtual machine 110 to continue operating during the process 700. At step 715 the data agent 155 calls an API function of the integration component 157 to unquiesce the file systems of the virtual machine 110. The data agent 155 may, subsequent to unquiescing the file systems in step 705, also unquiesce any applications that were previously quiesced.

At step 720 the data agent 155 determines (e.g., based upon the indication received in step 305 of the process 300 of FIG. 3), how to copy the data of the virtual machine 110. For example, the data agent 155 may copy the data of the virtual machine 110 in one of three ways: 1) a file-level copy; 2) an image-level copy; or 3) a disk-level copy.

File-Level Copy

If the indication specifies that the data agent 155 is to perform a file-level copy, the process 700 branches to the

file-level copy branch. For example, an administrator may provide that a file-level copy is to be performed if the administrator wishes to copy only certain files on a volume of a virtual disk 140 (e.g., only files within a certain directory or files that satisfy certain criteria). At step 722, the data agent 155 determines a mount point of the data store 165 on the virtual machine storage manager 145 (e.g., by dynamically determining an available mount point or by reading a stored indication of a mount point to use). For example, the mount point may be C:\mount\<virtual machine name>\ on the data store 165. At step 724 the data agent 155 determines the volumes of the virtual machine 110 (e.g., by calling an API function of the integration component 157 or by calling a function of the API component 210). For example, a virtual machine 110 using a Microsoft Windows operating system may have a C:\volume, a D:\volume, and so forth. At step 726 the data agent 155 mounts the determined volumes containing files at the determined mount point of the data store 165 (e.g., by again calling an API function of the integration component 157 or by calling a function of the API component 210).

As previously described, a virtual disk 140 corresponds to one or more files (e.g., one or more *.vmdk or *.vhd files), called virtual disk files, on the primary storage datastore 135. A volume may span one or more virtual disks 140, or one or more volumes may be contained within a virtual disk 140. When the data agent 155 mounts the determined volumes, the primary storage data store 135 sends to the VLUN driver 152 a block list of the virtual disk files corresponding to the virtual disks 140 of the determined volumes. The VLUN driver 152 uses the block list information to present the determined volumes (e.g., as read-only volumes or as read-write volumes) to the operating system of the virtual machine storage manager 145. The data agent 155 communicates with the VLUN driver 152 to mount the determined volumes at the mount point of the virtual machine storage manager 145. Using the previous examples of a virtual machine 110 with a C:\volume and a D:\volume, the data agent 155 would mount these volumes at the following respective locations:

C:\mount\<virtual machine name>\letters\C
C:\mount\<virtual machine name>\letters\D

After mounting the determined volumes, the data agent 155 can present to an administrator an interface displaying the mounted volumes and the directories and files on the mounted volumes, to enable the administrator to select which files and/or directories are to be copied. Alternatively, files and/or directories can be automatically selected in accordance with a storage policy determined by the virtual machine's 110 membership in a sub-client, or in accordance with a set of criteria or rules. At step 728 the data agent 155 copies the selected files and/or directories on the determined volumes to the secondary storage data store 175 (e.g., via a secondary storage computing device). The data agent 155 does so by providing an indication of a file and/or directory that is to be copied to the VLUN driver 152, which requests the blocks corresponding to the selected file and/or directory in the virtual disk files 140 on the primary storage datastore 135. The mapping between blocks and files/directories may be maintained by the primary storage data store 135 (e.g., in a table or other data structure).

After completing the copy, the data agent 155 at step 730 unmounts the determined volumes from the virtual machine storage manager 145 (e.g., by calling an API function of the

US 9,740,723 B2

19

integration component 157 or by calling a function of the API component 210). At step 732 the data agent 155 calls an API function of the integration component 157 to take the virtual machine 110 out of snapshot mode. Alternatively, the data agent 155 may call a function of the API component 210 to take the virtual machine 110 out of snapshot mode. Taking the virtual machine 110 out of snapshot mode consolidates the writes from the delta disk file (e.g., any intervening write operations to the virtual disk 140 between the time the virtual machine 110 was put into snapshot mode and the time it was taken out of snapshot mode) to the virtual disk file of the virtual disk 140. In this way, performing a copy operation on a primary copy of virtual machine data does not affect the virtual machine's 110 use of the data. Rather, operations can pick up at the point where they left off. The process 700 then concludes.

Volume-Level Copy

If the indication specifies that the data agent 155 is to perform a volume-level copy, the process 700 branches to the volume-level copy branch. The process for performing a volume-level copy is similar to that for performing a file-level copy, and steps 722 through 726 are effectively the same for this second branch of the process 700. At step 754 the data agent 155 analyzes the virtual volume and extracts metadata from the virtual volume. This process is described in more detail herein, e.g., with reference to FIG. 8.

After mounting the determined volumes (step 726 of the volume-level copy branch), the data agent 155 can present to an administrator an interface displaying the mounted volumes (optionally, the files and/or directories on the mounted volumes can also be displayed) to enable the administrator to select which volumes are to be copied. Alternatively, volumes can be automatically selected in accordance with a storage policy determined by the virtual machine's 110 membership in a sub-client, or in accordance with a set of criteria or rules. At step 734 the data agent 155 copies the selected volumes at the mount point on the virtual machine storage manager 145 to the secondary storage data store 175 (e.g., via a secondary storage computing device). The data agent 155 does so by providing an indication of a volume that is to be copied to the VLUN driver 152, which requests the blocks corresponding to the selected volumes in the virtual disk files 140 on the primary storage datastore 135. The mapping between blocks and volumes may be maintained by the primary storage data store 135 (e.g., in a table or other data structure).

After copying, the data agent 155 at step 730 unmounts the determined volumes from the virtual machine storage manager 145 (e.g., by calling an API function of the integration component 157 or by calling a function of the API component 210). At step 732 the data agent 155 calls an API function of the integration component 157 to take the virtual machine 110 out of snapshot mode. Alternatively, the data agent 155 may call a function of the API component 210 to take the virtual machine 110 out of snapshot mode. Taking the virtual machine 110 out of snapshot mode consolidates the writes to the delta disk file to the virtual disk file of the virtual disk 140. The process 700 then concludes.

One advantage of performing copy operations at the file-level or the volume-level is that the data agent 155 can copy the virtual machine data from the primary storage datastore 135 to the secondary storage data store 175 without having to copy it to the datastore 165 on the virtual machine storage manager 145. Stated another way, the data agent 155 can obtain the virtual machine data from the primary storage datastore 135, perform any specified operations upon it (e.g., compress it, single or variable instance it,

20

encrypt it, etc.), and stream the virtual machine data to the secondary storage data store 175 (e.g., via a secondary storage computing device 1065), without staging or caching the data at the virtual machine storage manager 145. This allows the data agent 155 to copy the data directly to the secondary storage data store 175, without first copying it to an intermediate location. Accordingly, the data agent 155 can quickly and efficiently perform file-level and volume-level copies of data of virtual machines 110.

Disk-Level Copy

The file-level copy and the volume-level copy can be thought of as operating at the virtual level. In other words, the data agent 155 may have to utilize data structures, functions or other information or aspects exposed or provided by a virtual machine 110 (or the virtual machine host 105) in order to copy the data of the virtual machine 110 to the secondary storage data store 175. For example, in order to perform a file-level or volume-level copy of the data of a virtual machine 110, the data agent 155 utilizes some information or aspect of the virtual machine 110 to determine its files and directories and/or volumes. The data agent 155 does so in order to present the determined files and directories or volumes for their selection by an administrator, or in order to apply, implement or execute storage operations according to a storage policy. In contrast, a disk-level copy can be thought of as operating at a non-virtual level (e.g., at a level of the physical computer hosting the virtual machine 110 and/or the physical storage media upon which the virtual machine data is stored). In other words, the data agent 155 can directly access the physical storage media storing the data of the virtual machine 110 (e.g., the primary storage data store 135 connected via the SAN 130) to copy the virtual disks 140 of the virtual machine 110. Because the data agent 155 is copying the virtual disks 140 without necessarily determining files and directories or volumes of the virtual machine 110, the data agent 155 does not necessarily have to utilize information or aspects of the virtual machine 110 or the virtual machine host 105.

If the indication specifies that the data agent 155 is to perform a disk-level copy, the process branches to the disk-level copy branch. At step 746, the data agent 155 determines a copy point on the data store 165. For example, the copy point may be C:\copy\<virtual machine name>-copy\virtualmachine\ on the data store 165. At step 748 the data agent 155 determines the virtual disk and any associated configuration files (e.g., the *.vmx file and/or the disk descriptor files) of the virtual machine 110 (e.g., by calling an API function of the integration component 157 or by calling a function of the API component 210). The primary storage data store 135 sends to the VLUN driver 152 a block list of the virtual disk and configuration files. At step 750, the data agent 155 copies these files to the copy point on the datastore 165 on the virtual machine storage manager 145. The data agent 155 does so by providing an indication of the virtual disk and configuration files to the VLUN driver 152, which requests the blocks corresponding to the virtual disk and configuration files from the primary storage datastore 135. The mapping between blocks and files/directories may be maintained by the primary storage data store 135 (e.g., in a table or other data structure).

At step 752 the data agent 155 calls an API function of the integration component 157 to take the virtual machine 110 out of snapshot mode (or calls a function of the API component 210). At step 754 the data agent 155 analyzes the virtual disk and configuration files and extracts metadata from the virtual disk and configuration files. This process is

US 9,740,723 B2

21

described in more detail herein, e.g., with reference to FIG. 8. At step 756 the data agent 155 copies the virtual disk and configuration files to the secondary storage data store 175. At step 758 the data agent 155 removes the copied virtual disk and configuration files from the data store 165 on the virtual machine storage manager 145. The process 700 then concludes.

Because a disk-level copy operates essentially at a non-virtual level, it may not have to utilize information or aspects of the virtual machine 110 (or the virtual machine host 105) in order to copy its data to the secondary storage data store 175. Therefore, a disk-level copy may not necessarily involve much of the overhead involved in a file-level copy or a volume-level copy. Rather, a disk-level copy can directly access the physical storage media storing the data of the virtual machine 110 (e.g., the primary storage data store 135) to copy the virtual disks 140 of the virtual machine 110. Because a disk-level copy can directly access the primary storage data store 135, the volumes on the virtual disks 140 do not need to be mounted. Accordingly, a disk-level copy may be performed faster and more efficiently than a file-level copy or a volume-level copy.

Process for Extracting Metadata

Certain steps in the following process for extracting metadata from the virtual volumes and/or the virtual disk and configuration files are described below using a configuration of a virtual machine 110 having a single virtual disk 140 comprised in a single virtual disk file. Those of skill in the art will understand that the process is not limited in any way to this configuration. Rather, the following process may be used to extract metadata from virtual disk and configuration files that are arranged or structured in a wide variety of configurations, such as multiple virtual disks 140 spanning multiple virtual disk files. Generally, metadata refers to data or information about data. Metadata may include, for example, data relating to relationships between virtual disk files, data relating to how volumes are structured on virtual disks 140, and data relating to a location of a file allocation table or a master file table. Metadata may also include data describing files and data objects (e.g., names of files or data objects, timestamps of files or data objects, ACL entries, and file or data object summary, author, source or other information). Metadata may also include data relating to storage operations or storage management, such as data locations, storage management components associated with data, storage devices used in performing storage operations, index data, data application type, or other data. Those of skill in the art will understand that metadata may include data or information about data other than the examples given herein.

FIG. 8 is a flow diagram illustrating a process 800 performed by the virtual disk analyzer component 160 for extracting metadata (e.g., file location metadata and metadata describing virtual disks 140 and/or files and/or volumes within virtual disks 140) from virtual volumes and/or virtual disk and configuration files. The process 800 begins at step 805, where the virtual disk analyzer component 160 of the data agent 155 accesses the configuration files to determine if there are any parent-child relationships between virtual disk files (e.g., the virtual disk analyzer component 160 determines how many links in a chain of virtual disk files there are). The virtual disk analyzer component 160 performs this step by reading and analyzing the virtual disk and/or configuration files.

For example, for a VMware virtual machine 110, the virtual disk analyzer component 160 may read and analyze the *.vmx configuration files and/or the *.vmdk disk descriptor files. In this example, the parent virtual disk 140

22

may be named “basedisk.vmdk.” The parent virtual disk 140 may have a *.vmdk disk descriptor file with an entry that uniquely identifies the parent virtual disk 140 having the following syntax:

[identifier-name]=[identifier-value]

For example, the entry CID=daf6cf10 comports with this syntax. A first child virtual disk 140 (e.g., a first snapshot) may be named “basedisk-000001.vmdk.” The first child virtual disk 140 may have a *.vmdk disk descriptor file with an entry that uniquely identifies its parent having the following syntax:

[parentidentifier-name]=[parentidentifier-value]

For example, the entry parentCID=daf6cf10 comports with this syntax. The virtual disk analyzer component 160 may identify parent-child relationships between virtual disk files in other ways, such as by observing access to virtual disk files and inferring the relationships by such observations. At step 810 the virtual disk analyzer component 160 determines the relationships between virtual disk files (the virtual disk analyzer component 160 determines how the virtual disk 140 is structured—how many extents make up each link in the chain). The virtual disk analyzer component 160 performs this step by reading and analyzing the disk descriptor file if it is a separate file or by reading the disk descriptor information if it is embedded into the virtual disk file. The virtual disk analyzer component 160 may determine the relationships between virtual disk files in other ways, such as by observing access to virtual disk files and inferring the relationships from such observations.

At step 815 the virtual disk analyzer component 160 determines how the partitions and volumes are structured on the virtual disks 140. The virtual disk analyzer component 160 does this by reading the sectors of the virtual disks 140 that contain the partition tables to determine how the virtual disk 140 is structured (e.g., whether it is a basic or a dynamic disk). The virtual disk analyzer component 160 also reads the sectors of the virtual disks 140 that contain the logical volume manager databases. Because the locations of these sectors (e.g., the sectors of the partition tables and the logical volume manager databases) are well-known and/or can be dynamically determined, the virtual disk analyzer component 160 can use techniques that are well-known to those of skill in the art to read those sectors and extract the necessary data. The virtual disk analyzer component 160 is thus able to determine how the virtual disks 140 is partitioned by the operating system 120 of the virtual machine 110 and how volumes are laid out in the virtual disks 140 (e.g., if there are simple volumes, spanned volumes, striped volumes, mirrored volumes, and/or RAID-5 volumes, etc.)

At step 820 the virtual disk analyzer component 160 determines the location of the Master File Table (MFT) or similar file allocation table for each volume. As with the partition tables and the logical volume manager databases, the locations of the sectors containing the MFT are well-known and/or can be dynamically determined. Therefore, the virtual disk analyzer component 160 can use techniques that are well-known to those of skill in the art to determine the location of the MFT. At step 825 the virtual disk analyzer component 160 stores the determined parent-child relationships and relationships between virtual disk files, the determined structure of volumes of the virtual disks 140, and the determined location of the MFT in a data structure, such as a table. For example, a table having the following schema may be used to store this information:

Virtual Machine ID	Virtual disk file relationships	Volume structures	Location of MFT
E.g., a substantially unique identifier for the virtual machine 110	E.g., description of the parent-child relationships, such as by a hierarchical description	E.g., partition information and how virtual volumes are laid out on virtual disks	E.g., the location of the MFT within each volume

The virtual disk analyzer component 160 may use other data structures to store this information in addition or as an alternative to the preceding table. The virtual disk analyzer component 160 may store this information in the secondary storage data store 175 or in another data store. The virtual disk analyzer component 160 may also collect other meta-data, such as metadata describing virtual disks 140 and/or metadata describing files and/or data objects within virtual disks 140. For example, instead of storing the determined location of the MFT, the virtual disk analyzer component 160 could store the locations of files or data objects within virtual disks 140. After storing this metadata, the process 800 then concludes.

Process for Restoring Data of Virtual Machines

FIG. 9 is a flow diagram illustrating a process 900 performed by the secondary storage computing device 1065 for restoring virtual machine data. One or more of the entities illustrated in the figures (e.g., FIGS. 1A, 1B, 2, and/or 10) may perform different aspects of the process 900. In some examples, an administrator at a management console instigates the process 900 by sending an indication to restore virtual machine data to the secondary storage computing device 1065. The secondary storage computing device 1065 accesses the index 1061 to locate the virtual machine data, and accesses the storage devices 1015 (e.g., the secondary storage data store 175) upon which the virtual machine data is located. The secondary storage computing device 1065 restores the data from the storage devices 1015 to a specified location (e.g., a location specified by the administrator).

The process 900 begins at step 905 where the secondary storage computing device 1065 receives an indication to restore data of one or more virtual machines 110. The indication can be to restore one or more files, one or more volumes, one or more virtual disks of a virtual machine 110, or an entire virtual machine 110. At step 910 the secondary storage computing device 1065 determines how (e.g., by analyzing the index 1061) the data agent 155 originally copied the virtual machine data, either: 1) a file-level copy; 2) an image-level copy; or 3) a disk-level copy.

Restore from a File-Level Copy

If the data agent 155 originally performed a file-level copy, the process 900 branches to the file-level restore branch. At step 915, the secondary storage computing device 1065 mounts a copy set corresponding to the files to be restored from the secondary storage data store 175. The copy set may be manually selected by an administrator or automatically selected based on an association between the copy set and the virtual machine from which the data in the copy set came. Additionally or alternatively, the copy set may be automatically determined based upon the metadata extracted and stored (described with reference to, e.g., FIG. 8) or based upon other metadata (e.g., metadata stored in index 1061).

Because the data agent 155 originally performed a file-level copy (of selected files and/or directories), the secondary storage computing device 1065 generally restores files

and/or directories out of the copy set. At step 920 the secondary storage computing device 1065 restores one or more files or directories (e.g., a single file) out of the copy set. For example, the secondary storage computing device 1065 can call a function of an API exposed by a virtual machine 110 or its hosting virtual machine host 105 to restore the one or more files or directories to the virtual machine 110. As another example, the secondary storage computing device 1065 can copy the one or more files or directories to the primary storage data store 135. The secondary storage computing device 1065 can restore the one or more files or directories to the original virtual machine 110 from which they were originally copied, to a different virtual machine 110, to a non-virtual machine, and/or to another storage device 1015. The process 900 then concludes.

Restore from a Volume-Level Copy

If the data agent 155 originally performed a volume-level copy, the process 900 branches to the volume-level restore branch. At step 915, the secondary storage computing device 1065 mounts a copy set corresponding to the files or volumes to be restored from the secondary storage data store 175. The copy set may be manually selected by an administrator or automatically selected based on an association between the copy set and the virtual machine 110 from which the data in the copy set came. Additionally or alternatively, the copy set may be automatically determined based upon the metadata extracted and stored (described with reference to, e.g., FIG. 8) or based upon other metadata (e.g., metadata stored in index 1061).

At step 945 the secondary storage computing device 1065 accesses metadata corresponding to the data that is to be restored (e.g., the determined location of the MFT). This is the metadata that was stored in step 825 of the process 800. At step 960 the secondary storage computing device 1065 uses the determined location of the MFT to access the MFT and use the entries in the MFT to determine where the files and directories on the virtual disk 140 are located (e.g., on which sectors of the virtual disk 140 a particular file is located).

Because the data agent 155 originally performed a volume-level copy (of selected volumes including files and/or directories within the volumes), the secondary storage computing device 1065 can generally restore both files and/or directories and entire volumes (e.g., an entire C:\volume, an entire D:\volume, etc.) out of the copy set. If the secondary storage computing device 1065 is to restore a file, the process 900 branches to step 920. At this step the secondary storage computing device 1065 restores one or more files or directories out of the copy set (e.g., a single file). The secondary storage computing device 1065 can restore the one or more files or directories to the original virtual machine 110 from which they were originally copied, to a different virtual machine 110, to a non-virtual machine, and/or to another storage device 1015. For example, if the original virtual machine 110 no longer exists, the one or more files or directories may be restored to its replacement.

If instead, the secondary storage computing device 1065 is to restore a volume, the process 900 branches to step 930. At this step the secondary storage computing device 1065 restores one or more volumes out of the copy set. The data agent 155 secondary storage computing device 1065 can restore the one or more volumes to the original virtual machine 110 from which they were originally copied up, to a different virtual machine 110, or to a non-virtual machine and/or to another storage device 1015. For example, a C:\volume may be restored out of a copy set to the original virtual machine 110 from which it was copied, thus over-

US 9,740,723 B2

25

writing its existing C:\volume. As another example, a D:\volume may be restored out of a copy set to another virtual machine 110, thus replacing its current D:\volume.

The secondary storage computing device 1065 may restore the files, directories and/or volumes to various locations. For example, the secondary storage computing device 1065 can copy the files, directories and/or volumes to the primary storage data store 135. The secondary storage computing device 1065 can restore the one or more volumes to the original virtual machine 110 from which they were originally copied up, to a different virtual machine 110, to a non-virtual machine (e.g., to a physical machine), and/or to another storage device 1015. For example, an entire D:\volume from an original virtual machine 110 may be restored to the original virtual machine 110, to another virtual machine 110 and/or to a non-virtual machine (e.g., to a physical machine). As described in more detail herein, a volume of a virtual machine 110 may be restored in its original format (e.g., if the volume came from a VMware virtual machine 110, it can be restored as a volume in the VMware format, such as a *.vmdk file) or converted to another format (e.g., if the volume came from a VMware virtual machine 110, it can be restored as a volume in the Microsoft format, such as a *.vhd file). The secondary storage computing device 1065 can also restore the volume as a container file, from which the volume can be extracted. After steps 920 and/or 930, the process 900 then concludes. Restore from a Disk-Level Copy

If the data agent 155 originally performed a disk-level copy, the process 900 branches to the disk-level restore branch. At step 915, the secondary storage computing device 1065 mounts a copy set corresponding to the virtual disks, files, volumes, and/or virtual machines 110 to be restored from the secondary storage data store 175. The copy set may be manually selected by an administrator or automatically selected based on an association between the copy set and the virtual machine from which the data in the copy set came. Additionally or alternatively, the copy set may be automatically determined based upon the metadata extracted and stored (described herein, e.g., with reference to FIG. 8) or based upon other metadata (e.g., metadata stored in index 1061).

At step 945 the secondary storage computing device 1065 accesses metadata corresponding to the data that is to be restored (e.g., the determined parent-child relationships and relationships between virtual disk files, the determined structure of volumes of the virtual disks 140, and the determined location of the MFT). This is the metadata that was stored in step 825 of the process 800. At step 950 the secondary storage computing device 1065 uses the determined parent-child relationships and relationships between virtual disk files to reconstruct the virtual disks 140. For example, if a virtual disk 140 is comprised of numerous virtual disk files, the secondary storage computing device 1065 uses the determined relationships between them to link them together into a single virtual disk file. In so doing, the secondary storage computing device 1065 may access grain directories and grain tables within virtual disk files. Grain directories and grain tables are data structures located within virtual disk files that specify the sectors (blocks) within virtual disks 140 that have been allocated for data storage. The secondary storage computing device 1065 may access these data structures to locate data within virtual disks 140.

At step 955 the secondary storage computing device 1065 uses the determined structure of volumes of the virtual disks 140 to reconstruct the volumes. At step 960 the secondary storage computing device 1065 uses the determined location

26

of the MFT to access the MFT and uses the entries in the MFT to determine where the files and directories on the virtual disk 140 are located (e.g., on which sectors of the virtual disk 140 a particular file is located).

Because the data agent 155 originally performed a disk-level copy (of virtual disk and configuration files), the secondary storage computing device 1065 can restore files or directories, entire volumes (e.g., an entire C:\volume, an entire D:\volume, etc.) as well as an entire virtual machine 110 out of the copy set. If an entire virtual machine 110 is to be restored, the process 900 branches to step 965. The secondary storage computing device 1065 can copy all the virtual disk and configuration files to the location where the entire virtual machine 110 is to be restored. This can be the original location of the virtual machine 110 (on the original virtual machine host 105), or it can be a new location where the virtual machine had not originally been located (e.g., on a new virtual machine host 105.) If the virtual disk and configuration files are copied to the original virtual machine host 105, the virtual machine host 105 should be able to restart the virtual machine 110, which can then recommence operating in the state it existed in when its virtual disk and configuration files were originally copied.

Similarly, if the virtual disk and configuration files are copied to a new virtual machine host 105, the new virtual machine host 105 should be able to start the virtual machine 110, which can then commence operating in the state it existed in when its virtual disk and configuration files were originally copied. The ability to restore a virtual machine 110 to a new virtual machine host 105 other than its original virtual machine host 105 allows virtual machines 110 to be moved or “floated” from one virtual machine host 105 to another. The secondary storage computing device 1065 can also restore the entire virtual machine 110 as a container file, from which the entire virtual machine 110 can be extracted. After step 965, the process 900 then concludes.

If instead of restoring an entire virtual machine 110, the secondary storage computing device 1065 is to restore a volume, the process 900 branches to step 930. At this step the secondary storage computing device 1065 restores one or more volumes out of the copy set. After step 930, the process 900 then concludes.

If instead of restoring an entire virtual machine 110 or a volume, the secondary storage computing device 1065 is to restore a file, the process 900 branches to step 920. At this step the secondary storage computing device 1065 restores one or more files or directories out of the copy set (e.g., a single file). The secondary storage computing device 1065 can restore the one or more files or directories to the original virtual machine 110 from which they were originally copied, to a different virtual machine 110, to non-virtual machine, and/or to another storage device 1015. The process 900 then concludes.

If instead of restoring an entire virtual machine 110, a volume, or a file, the secondary storage computing device 1065 is to restore one or more virtual disks, the process 900 branches to step 970. At this step the secondary storage computing device 1065 restores the virtual disk and configuration files corresponding to the one or more virtual disks to be restored out of the copy set. The secondary storage computing device 1065 can restore the one or more virtual disks to the original virtual machine host 105 from which they were originally copied. Additionally or alternatively, the secondary storage computing device 1065 can restore the one or more virtual disks to the original virtual machine 110 from which they were originally copied, to a different virtual machine 110, to a non-virtual machine,

US 9,740,723 B2

27

and/or to another storage device **1015**. If the one or more virtual disks are to be restored to the virtual machine **105**, they may overwrite, replace and/or supplement the existing virtual disks of a virtual machine **110**. The process **900** then concludes.

Depending upon what the secondary storage computing device **1065** is to restore, certain steps in the process **900** may not need to be performed. For example, if the secondary storage computing device **1065** is to restore an entire virtual machine **110** out of a disk-level copy, the data agent **155** may not need to access the stored metadata (step **945**) or reconstruct the virtual disk **140**, volumes and files (steps **950**, **955** and **960**). The data agent **155** can simply mount the copy set and copy the virtual disk and configuration files to the appropriate location. As another example, if the secondary storage computing device **1065** is to restore a volume out of a disk-level copy, the secondary storage computing device **1065** may not need to reconstruct files using the MFT, as mentioned above. The secondary storage computing device **1065** can simply reconstruct the volumes and then copy the volumes to the appropriate location. Those of skill in the art will understand that more or fewer steps than those illustrated in the process **900** may be used to restore data of virtual machines **110**.

As previously described, one advantage of performing a disk-level copy is that it may be quicker and more efficient than file-level or volume-level copying. Also as previously described, the process of extracting metadata from the virtual disk and configuration files enables the ability to restore individual files, directories and/or volumes to the virtual machine **110** or to other locations (e.g., to other virtual machines **110** to non-virtual machines, and/or to other storage devices **1015**). The combination of a disk-level copy and the capability to restore individual files, directories and/or volumes of a virtual machine **110** provides for a fast and efficient process for duplicating primary copies of data, while still enabling granular access (e.g., at the individual file or data object level) to the duplicated primary data (granular access to the secondary copies of data is enabled). This combination optimizes the aspect of virtual machine data management that is likely performed most frequently (duplication of primary copies of data), but not at the expense of the aspect that is likely performed less often (restoration of secondary copies of data), because granular access to duplicated primary copies of data is still enabled. Suitable Data Storage Enterprise

FIG. **10** illustrates an example of one arrangement of resources in a computing network, comprising a data storage system **1050**. The resources in the data storage system **1050** may employ the processes and techniques described herein. The system **1050** includes a storage manager **1005**, one or more data agents **1095**, one or more secondary storage computing devices **1065**, one or more storage devices **1015**, one or more clients **1030**, one or more data or information stores **1060** and **1062**, a single instancing database **1023**, an index **1011**, a jobs agent **1020**, an interface agent **1025**, and a management agent **1031**. The system **1050** may represent a modular storage system such as the CommVault Qinetix system, and also the CommVault GALAXY backup system, available from CommVault Systems, Inc. of Oceanport, N.J., aspects of which are further described in the commonly-assigned U.S. patent application Ser. No. 09/610,738, now U.S. Pat. No. 7,035,880, the entirety of which is incorporated by reference herein. The system **1050** may also represent a modular storage system such as the CommVault Simpana system, also available from CommVault Systems, Inc.

28

The system **1050** may generally include combinations of hardware and software components associated with performing storage operations on electronic data. Storage operations include copying, backing up, creating, storing, retrieving, and/or migrating primary storage data (e.g., data stores **1060** and/or **1062**) and secondary storage data (which may include, for example, snapshot copies, backup copies, HSM copies, archive copies, and other types of copies of electronic data stored on storage devices **1015**). The system **1050** may provide one or more integrated management consoles for users or system processes to interface with in order to perform certain storage operations on electronic data as further described herein. Such integrated management consoles may be displayed at a central control facility or several similar consoles distributed throughout multiple network locations to provide global or geographically specific network data storage information.

In one example, storage operations may be performed according to various storage preferences, for example, as expressed by a user preference, a storage policy, a schedule policy, and/or a retention policy. A "storage policy" is generally a data structure or other information source that includes a set of preferences and other storage criteria associated with performing a storage operation. The preferences and storage criteria may include, but are not limited to, a storage location, relationships between system components, network pathways to utilize in a storage operation, data characteristics, compression or encryption requirements, preferred system components to utilize in a storage operation, a single instancing or variable instancing policy to apply to the data, and/or other criteria relating to a storage operation. For example, a storage policy may indicate that certain data is to be stored in the storage device **1015**, retained for a specified period of time before being aged to another tier of secondary storage, copied to the storage device **1015** using a specified number of data streams, etc.

A "schedule policy" may specify a frequency with which to perform storage operations and a window of time within which to perform them. For example, a schedule policy may specify that a storage operation is to be performed every Saturday morning from 2:00 a.m. to 4:00 a.m. In some cases, the storage policy includes information generally specified by the schedule policy. (Put another way, the storage policy includes the schedule policy.) Storage policies and/or schedule policies may be stored in a database of the storage manager **1005**, to archive media as metadata for use in restore operations or other storage operations, or to other locations or components of the system **1050**.

The system **1050** may comprise a storage operation cell that is one of multiple storage operation cells arranged in a hierarchy or other organization. Storage operation cells may be related to backup cells and provide some or all of the functionality of backup cells as described in the assignee's U.S. patent application Ser. No. 09/354,058, now U.S. Pat. No. 7,395,282, which is incorporated herein by reference in its entirety. However, storage operation cells may also perform additional types of storage operations and other types of storage management functions that are not generally offered by backup cells.

Storage operation cells may contain not only physical devices, but also may represent logical concepts, organizations, and hierarchies. For example, a first storage operation cell may be configured to perform a first type of storage operations such as HSM operations, which may include backup or other types of data migration, and may include a variety of physical components including a storage manager **1005** (or management agent **1031**), a secondary storage

US 9,740,723 B2

29

computing device **1065**, a client **1030**, and other components as described herein. A second storage operation cell may contain the same or similar physical components; however, it may be configured to perform a second type of storage operations, such as storage resource management (“SRM”) operations, and may include monitoring a primary data copy or performing other known SRM operations.

Thus, as can be seen from the above, although the first and second storage operation cells are logically distinct entities configured to perform different management functions (HSM and SRM, respectively), each storage operation cell may contain the same or similar physical devices. Alternatively, different storage operation cells may contain some of the same physical devices and not others. For example, a storage operation cell configured to perform SRM tasks may contain a secondary storage computing device **1065**, client **1030**, or other network device connected to a primary storage volume, while a storage operation cell configured to perform HSM tasks may instead include a secondary storage computing device **1065**, client **1030**, or other network device connected to a secondary storage volume and not contain the elements or components associated with and including the primary storage volume. (The term “connected” as used herein does not necessarily require a physical connection; rather, it could refer to two devices that are operably coupled to each other, communicably coupled to each other, in communication with each other, or more generally, refer to the capability of two devices to communicate with each other.) These two storage operation cells, however, may each include a different storage manager **1005** that coordinates storage operations via the same secondary storage computing devices **1065** and storage devices **1015**. This “overlapping” configuration allows storage resources to be accessed by more than one storage manager **1005**, such that multiple paths exist to each storage device **1015** facilitating failover, load balancing, and promoting robust data access via alternative routes.

Alternatively or additionally, the same storage manager **1005** may control two or more storage operation cells (whether or not each storage operation cell has its own dedicated storage manager **1005**). Moreover, in certain embodiments, the extent or type of overlap may be user-defined (through a control console) or may be automatically configured to optimize data storage and/or retrieval.

Data agent **1095** may be a software module or part of a software module that is generally responsible for performing storage operations on the data of the client **1030** stored in data store **1060/1062** or other memory location. Each client **1030** may have at least one data agent **1095** and the system **1050** can support multiple clients **1030**. Data agent **1095** may be distributed between client **1030** and storage manager **1005** (and any other intermediate components), or it may be deployed from a remote location or its functions approximated by a remote process that performs some or all of the functions of data agent **1095**.

The overall system **1050** may employ multiple data agents **1095**, each of which may perform storage operations on data associated with a different application. For example, different individual data agents **1095** may be designed to handle Microsoft Exchange data, Lotus Notes data, Microsoft Windows 2000 file system data, Microsoft Active Directory Objects data, and other types of data known in the art. Other embodiments may employ one or more generic data agents **1095** that can handle and process multiple data types rather than using the specialized data agents described above.

If a client **1030** has two or more types of data, one data agent **1095** may be required for each data type to perform

30

storage operations on the data of the client **1030**. For example, to back up, migrate, and restore all the data on a Microsoft Exchange 2000 server, the client **1030** may use one Microsoft Exchange 2000 Mailbox data agent **1095** to back up the Exchange 2000 mailboxes, one Microsoft Exchange 2000 Database data agent **1095** to back up the Exchange 2000 databases, one Microsoft Exchange 2000 Public Folder data agent **1095** to back up the Exchange 2000 Public Folders, and one Microsoft Windows 2000 File System data agent **1095** to back up the file system of the client **1030**. These data agents **1095** would be treated as four separate data agents **1095** by the system even though they reside on the same client **1030**.

Alternatively, the overall system **1050** may use one or more generic data agents **1095**, each of which may be capable of handling two or more data types. For example, one generic data agent **1095** may be used to back up, migrate and restore Microsoft Exchange 2000 Mailbox data and Microsoft Exchange 2000 Database data while another generic data agent **1095** may handle Microsoft Exchange 2000 Public Folder data and Microsoft Windows 2000 File System data, etc.

Data agents **1095** may be responsible for arranging or packing data to be copied or migrated into a certain format such as an archive file. Nonetheless, it will be understood that this represents only one example, and any suitable packing or containerization technique or transfer methodology may be used if desired. Such an archive file may include metadata, a list of files or data objects copied, the file, and data objects themselves. Moreover, any data moved by the data agents may be tracked within the system by updating indexes associated with appropriate storage managers **1005** or secondary storage computing devices **1065**. As used herein, a file or a data object refers to any collection or grouping of bytes of data that can be viewed as one or more logical units.

Generally speaking, storage manager **1005** may be a software module or other application that coordinates and controls storage operations performed by the system **1050**. Storage manager **1005** may communicate with some or all elements of the system **1050**, including clients **1030**, data agents **1095**, secondary storage computing devices **1065**, and storage devices **1015**, to initiate and manage storage operations (e.g., backups, migrations, data recovery operations, etc.).

Storage manager **1005** may include a jobs agent **1020** that monitors the status of some or all storage operations previously performed, currently being performed, or scheduled to be performed by the system **1050**. Jobs agent **1020** may be communicatively coupled to an interface agent **1025** (e.g., a software module or application). Interface agent **1025** may include information processing and display software, such as a graphical user interface (“GUI”), an application programming interface (“API”), or other interactive interface through which users and system processes can retrieve information about the status of storage operations. For example, in an arrangement of multiple storage operations cell, through interface agent **1025**, users may optionally issue instructions to various storage operation cells regarding performance of the storage operations as described and contemplated herein. For example, a user may modify a schedule concerning the number of pending snapshot copies or other types of copies scheduled as needed to suit particular needs or requirements. As another example, a user may employ the GUI to view the status of pending storage operations in some or all of the storage operation cells in a given network or to monitor the status of certain components

US 9,740,723 B2

31

in a particular storage operation cell (e.g., the amount of storage capacity left in a particular storage device **1015**).

Storage manager **1005** may also include a management agent **1031** that is typically implemented as a software module or application program. In general, management agent **1031** provides an interface that allows various management agents **1031** in other storage operation cells to communicate with one another. For example, assume a certain network configuration includes multiple storage operation cells hierarchically arranged or otherwise logically related in a WAN or LAN configuration. With this arrangement, each storage operation cell may be connected to the other through each respective interface agent **1025**. This allows each storage operation cell to send and receive certain pertinent information from other storage operation cells, including status information, routing information, information regarding capacity and utilization, etc. These communications paths may also be used to convey information and instructions regarding storage operations.

For example, a management agent **1031** in a first storage operation cell may communicate with a management agent **1031** in a second storage operation cell regarding the status of storage operations in the second storage operation cell. Another illustrative example includes the case where a management agent **1031** in a first storage operation cell communicates with a management agent **1031** in a second storage operation cell to control storage manager **1005** (and other components) of the second storage operation cell via management agent **1031** contained in storage manager **1005**.

Another illustrative example is the case where management agent **1031** in a first storage operation cell communicates directly with and controls the components in a second storage operation cell and bypasses the storage manager **1005** in the second storage operation cell. If desired, storage operation cells can also be organized hierarchically such that hierarchically superior cells control or pass information to hierarchically subordinate cells or vice versa.

Storage manager **1005** may also maintain an index, a database, or other data structure **1011**. The data stored in database **1011** may be used to indicate logical associations between components of the system, user preferences, management tasks, media containerization and data storage information or other useful data. For example, the storage manager **1005** may use data from database **1011** to track logical associations between secondary storage computing device **1065** and storage devices **1015** (or movement of data as containerized from primary to secondary storage).

Generally speaking, the secondary storage computing device **1065**, which may also be referred to as a media agent, may be implemented as a software module that conveys data, as directed by storage manager **1005**, between a client **1030** and one or more storage devices **1015** such as a tape library, a magnetic media storage device, an optical media storage device, or any other suitable storage device. In one embodiment, secondary storage computing device **1065** may be communicatively coupled to and control a storage device **1015**. A secondary storage computing device **1065** may be considered to be associated with a particular storage device **1015** if that secondary storage computing device **1065** is capable of routing and storing data to that particular storage device **1015**.

In operation, a secondary storage computing device **1065** associated with a particular storage device **1015** may instruct the storage device to use a robotic arm or other retrieval means to load or eject a certain storage media, and to subsequently archive, migrate, or restore data to or from that media. Secondary storage computing device **1065** may

32

communicate with a storage device **1015** via a suitable communications path such as a SCSI or Fibre Channel communications link. In some embodiments, the storage device **1015** may be communicatively coupled to the storage manager **1005** via a SAN.

Each secondary storage computing device **1065** may maintain an index, a database, or other data structure **1061** that may store index data generated during storage operations for secondary storage (SS) as described herein, including creating a metabase (MB). For example, performing storage operations on Microsoft Exchange data may generate index data. Such index data provides a secondary storage computing device **1065** or other external device with a fast and efficient mechanism for locating data stored or backed up. Thus, a secondary storage computing device index **1061**, or a database **1011** of a storage manager **1005**, may store data associating a client **1030** with a particular secondary storage computing device **1065** or storage device **1015**, for example, as specified in a storage policy, while a database or other data structure in secondary storage computing device **1065** may indicate where specifically the data of the client **1030** is stored in storage device **1015**, what specific files were stored, and other information associated with storage of the data of the client **1030**. In some embodiments, such index data may be stored along with the data backed up in a storage device **1015**, with an additional copy of the index data written to index cache in a secondary storage device. Thus the data is readily available for use in storage operations and other activities without having to be first retrieved from the storage device **1015**.

Generally speaking, information stored in cache is typically recent information that reflects certain particulars about operations that have recently occurred. After a certain period of time, this information is sent to secondary storage and tracked. This information may need to be retrieved and uploaded back into a cache or other memory in a secondary computing device before data can be retrieved from storage device **1015**. In some embodiments, the cached information may include information regarding format or containerization of archives or other files stored on storage device **1015**.

One or more of the secondary storage computing devices **1065** may also maintain one or more single instance databases **1023**. Single instancing (alternatively called data deduplication) generally refers to storing in secondary storage only a single instance of each data object (or data block) in a set of data (e.g., primary data). More details as to single instancing may be found in one or more of the following previously-referenced U.S. patent application Ser. Nos. 11/269,512, 12/145,347, 12/145,342, 11/963,623, 11/950,376, and 61/100,686.

In some examples, the secondary storage computing devices **1065** maintain one or more variable instance databases. Variable instancing generally refers to storing in secondary storage one or more instances, but fewer than the total number of instances, of each data object (or data block) in a set of data (e.g., primary data). More details as to variable instancing may be found in the previously-referenced U.S. Pat. App. No. 61/164,803.

In some embodiments, certain components may reside and execute on the same computer. For example, in some embodiments, a client **1030** such as a data agent **1095**, or a storage manager **1005**, coordinates and directs local archiving, migration, and retrieval application functions as further described in the previously-referenced U.S. patent application Ser. No. 09/610,738. This client **1030** can function independently or together with other similar clients **1030**.

US 9,740,723 B2

33

As shown in FIG. 10, secondary storage computing devices **1065** each has its own associated metabase **1061**. Each client **1030** may also have its own associated metabase **1070**. However in some embodiments, each “tier” of storage, such as primary storage, secondary storage, tertiary storage, etc., may have multiple metabases or a centralized metabase, as described herein. For example, rather than a separate metabase or index associated with each client **1030** in FIG. 10, the metabases on this storage tier may be centralized. Similarly, second and other tiers of storage may have either centralized or distributed metabases. Moreover, mixed architecture systems may be used if desired, that may include a first tier centralized metabase system coupled to a second tier storage system having distributed metabases and vice versa, etc.

Moreover, in operation, a storage manager **1005** or other management module may keep track of certain information that allows the storage manager **1005** to select, designate, or otherwise identify metabases to be searched in response to certain queries as further described herein. Movement of data between primary and secondary storage may also involve movement of associated metadata and other tracking information as further described herein.

In some examples, primary data may be organized into one or more sub-clients. A sub-client is a portion of the data of one or more clients **1030**, and can contain either all of the data of the clients **1030** or a designated subset thereof. As depicted in FIG. 10, the data store **1062** includes two sub-clients. For example, an administrator (or other user with the appropriate permissions; the term administrator is used herein for brevity) may find it preferable to separate email data from financial data using two different sub-clients having different storage preferences, retention criteria, etc. Detection of Virtual Machines and Other Virtual Resources

As previously noted, because virtual machines **110** may be easily set up and torn down, they may be less permanent in nature than non-virtual machines. Due to this potential transience of virtual machines **110**, it may be more difficult to detect them, especially in a heterogeneous or otherwise disparate environment. For example, a virtual machine host **105** may host a number of different virtual machines **110**. Virtual machines **110** may be discovered using the techniques previously described herein. Alternatively or additionally, virtual machines **110** could be detected by periodically performing dynamic virtual resource detection routines to identify virtual machines **110** in the network **180** (or some subset thereof, such as a subnet). For example, the data agent **155** (or other agent) could analyze program behaviors corresponding to known virtual resource behaviors, perform fingerprint, hash, or other characteristic-based detection methods or routines, query a system datastore (e.g., the Windows registry) or other data structure of the virtual machine host **105** for keys or other identifiers associated with virtual resources. The data agent **155** may use other methods and/or combinations of these methods to detect virtual machines **110**.

Once detected, the data agent **155** could maintain virtual machine identifiers in a database or other data structure and use associated program logic to track existing virtual machines **110** in the network **180**. Alternatively or additionally, an administrator could manually populate the database, or it could be populated as part of an install or virtual resource creation process, or by an agent or other software module directed to detecting installation of virtual machines. The data agent **155** could update the database to remove a virtual machine identifier upon receiving an affirmative indication that the corresponding virtual machine **110** has

34

been taken down or removed from its virtual machine host **105**. Alternatively or additionally, the data agent **155** could periodically poll virtual machines **110** to determine if the virtual machines **110** are still functioning. If a virtual machine **110** does not respond after a certain number of polling attempts, the data agent **155** may assume that the virtual machine **110** is no longer functioning and thus remove its identifier from the database. Alternatively or additionally, the virtual machines **110** could periodically notify the data agent **155** that they are still functioning (e.g., by sending heartbeat messages to the data agent **155**). Upon a failure to receive notifications from a virtual machine **110** within a certain time period, the data agent **155** could remove its identifier from the database. The data agent **155** may use other methods and/or combinations of these methods to maintain an up-to-date listing of virtual machine identifiers in the database.

These techniques for detecting virtual machines **110** and maintaining identifiers thereof may also be used to detect virtual resources of virtual machines **110** and maintain identifiers thereof. For example, a virtual machine **110** may be coupled to a virtual storage device such as a virtual NAS device or a virtual optical drive. The data agent **155** could detect these virtual resources and maintain identifiers for them in a database or other data structure. The virtual resources may then be addressed as if they were actual resources. Once detected or identified, storage operations related to the virtual resources could be performed according to non-virtualized storage policies or preferences, according to storage policies or preferences directed specifically to virtual resources, and/or to combinations of non-virtualized and virtualized storage policies and preferences. As another example, a virtual machine **110** may be coupled to a virtual tape library (VTL). The data agent **155** may perform additional analysis on the nature and structure of the virtual resource which underlies the VTL (e.g., a virtual disk **140**). This may allow the data agent **155** to realize additional optimizations relating to storage operations associated with the data of the VTL. For example, even though the virtual resource is a VTL (necessitating sequential access), storage operations might be able to be performed non-linearly or in a random access fashion since the underlying virtual resource allows random access. Therefore, rather than sequentially seeking through the VTL data to arrive at a particular point, the data agent **155** could simply go directly to the relevant data on the virtual disk **140** that is the subject of the storage operation.

Indexing Virtual Machine Data

In traditional copy or backup of virtual machines **110**, an indexing agent is typically located at each virtual machine **110** or is otherwise associated with each virtual machine **110**. The indexing agent indexes data on the virtual machine **110**. This results in the creation of one index per virtual machine **110**. This facilitates searching of data on a per virtual machine **110** basis, but may make it difficult to search data across multiple virtual machines **110**. Moreover, the indexing is performed on the virtual machine **110** and thus uses its resources, which may not be desirable.

In contrast, copying of data of virtual machines **110** using the techniques described herein may use one indexing agent that is associated with multiple virtual machines **110**. The sole indexing agent thus indexes multiple virtual machines **110**. This results in the creation of one index for the multiple virtual machines **110**. The one indexing agent can subdivide or logically separate the single index into multiple sub-indexes for each virtual machine **110**. This technique facilitates searching of data using one index across multiple

US 9,740,723 B2

35

virtual machines **110** and also allows searching on a per virtual machine **110** basis. The sole indexing agent may create the single index using secondary copies of virtual machine data so as not to impact the primary copies or utilize virtual machine resources. The indexed data may be tagged by users. More details as to indexing data are described in the previously-referenced U.S. patent application Ser. No. 11/694,869.

Classification of Virtual Machine Data

As shown in FIG. **10**, clients **1030** and secondary storage computing devices **1065** may each have associated metabas- (10) es (**1070** and **1061**, respectively). Each virtual machine **110** may also have its own metabase containing metadata about virtual machine data. Alternatively, one or more virtual machines **110** may be associated with one or more metabas- (15) es. A classification agent may analyze virtual machines **110** to identify data objects or other files, email, or other information currently stored or present by the virtual machines **110** and obtain certain information regarding the information, such as any available metadata. Such metadata may include information about data objects or characteristics associated with data objects, such as data owner (e.g., the client or user that generates the data or other data manager), last modified time (e.g., the time of the most recent modification), data size (e.g., number of bytes of data), information about the data content (e.g., the applica- (20) tion that generated the data, the user that generated the data, etc.), to/from information for email (e.g., an email sender, recipient, or individual or group on an email distribution list), creation date (e.g., the date on which the data object was created), file type (e.g., the format or application type), last accessed time (e.g., the time the data object was most recently accessed or viewed), application type (e.g., the application that generated the data object), location/network (e.g., a current, past, or future location of the data object and network pathways to/from the data object), frequency of change (e.g., a period in which the data object is modified), business unit (e.g., a group or department that generates, manages, or is otherwise associated with the data object), and aging information (e.g., a schedule, which may include a time period, in which the data object is migrated to secondary or long-term storage), etc. The information obtained in this analyzing process may be used to initially create or populate the metabas- (25)

Alternatively or additionally, a journaling agent may populate the metabase with content by accessing virtual machines **110**, or by directly accessing virtual resources (e.g., virtual disks **140**). The journaling agent may include a virtual filter driver program and may be deployed on a virtual input/output port or data stack and operate in conjunction with a virtual file management program to record a virtual machine's interactions with its virtual data. This may involve creating a data structure such as a record or journal of each interaction. The records may be stored in a journal data structure and may chronicle data interactions on an interaction-by-interaction basis. The journal may include information regarding the type of interaction that has occurred along with certain relevant properties of the data involved in the interaction. The classification agent may analyze and process entries within respective journals associated with journaling agents, and report results to the metabase. More details as to techniques used in the classification of data and journaling of changes to data may be found in the previously-referenced U.S. patent application Ser. No. 11/564,119. (30)

36

Searching Virtual Machine Data

Once virtual machine data has been indexed and/or classified, users can search for virtual machine data using techniques known to those of skill in the art. The system may provide a single interface directed to enabling the search for virtual machine data (as well as non-virtual machine data). A user can utilize the interface to provide a query which is used to search metabas- (5) es and/or indices of virtual machine data (as well as non-virtual machine data). The system can in return provide results from the metabas- (10) es and/or indices relevant to the query that may be segregated based upon their origin (e.g., based upon whether they came from virtual machines or non-virtual machines). The returned results may be optionally analyzed for relevance, arranged, and placed in a format suitable for subsequent use (e.g., with another application), or suitable for viewing by a user and reported. More details as to techniques for searching data and providing results may be found in commonly-assigned U.S. patent application Ser. No. 11/931,034 (entitled METHOD AND SYSTEM FOR SEARCHING STORED DATA), the entirety of which is incorporated by reference herein. (15)

Single or Variable Instancing Virtual Machine Data

Virtual machine data may be single or variable instanced or de-duplicated in order to reduce the number of instances of stored data, sometimes to as few as one. For example, a virtual machine host **105** may host numerous virtual machines **110** configured identically or with slight variations (e.g., the virtual machines have the same operating system files, but different application data files). As another example, a virtual machine **110** may store substantially the same data in a virtual disk **140** that a non-virtual machine stores on its storage devices (e.g., both a virtual machine **110** and a non-virtual machine may have a C:\Windows directory and corresponding system files, and only one instance of each system file may need to be stored). If only a single instance of each data object in this data (the data of both the virtual machines and the non-virtual machines) can be stored on a single instance storage device, significant savings in storage space may be realized. (20)

To single or variable instance virtual machine data, an agent (e.g., a media agent) may generate a substantially unique identifier (for example, a hash value, message digest, checksum, digital fingerprint, digital signature or other sequence of bytes that substantially uniquely identifies the file or data object) for each virtual data object. The word "substantially" is used to modify the term "unique identifier" because algorithms used to produce hash values may result in collisions, where two different files or data objects result in the same hash value. However, depending upon the algorithm or cryptographic hash function used, collisions should be suitably rare and thus the identifier generated for a virtual file or data object should be unique throughout the system. (25)

After generating the substantially unique identifier for the virtual data object, the agent determines whether it should be stored on the single instance storage device. To determine this, the agent accesses a single instance database to determine if a copy or instance of the data object has already been stored on the single instance storage device. The single instance database utilizes one or more tables or other data structures to store the substantially unique identifiers of the data objects that have already been stored on the single instance storage device. If a copy or instance of the data object has not already been stored on single instance storage device, the agent sends the copy of the virtual data object to the single instance storage device for storage and adds its substantially unique identifier to the single instance data- (30)

US 9,740,723 B2

37

base. If a copy or instance of the data object has already been stored, the agent can avoid sending another copy to the single instance storage device. In this case, the agent may add a reference (e.g., to an index in the single instance database, such as by incrementing a reference count in the index) to the already stored instance of the data object. Adding a reference to the already stored instance of the data object enables storing only a single instance of the data object while still keeping track of other instances of the data object that do not need to be stored.

Redundant instances of data objects may be detected and reduced at several locations or times throughout the operation of the system. For example, the agent may single or variable instance virtual machine data prior to performing any other storage operations. Alternatively or additionally, the agent may single instance virtual machine data after it has been copied to the secondary storage data store 175. The agent may generate a substantially unique identifier and send it across the network 180 to the single instance database to determine if the corresponding virtual data object should be stored, or the agent may send the virtual data object to the single instance database, which then may generate a substantially unique identifier for it. More details as to single instancing data may be found in one or more of the previously-referenced described in one or more of previously-referenced U.S. patent application Ser. Nos. 11/269,512, 12/145,347, 12/145,342, 11/963,623, 11/950,376, 61/100,686, and 61/164,803.

Protecting Virtual Machine Data in Homogenous and Heterogeneous Environments

The techniques described herein are applicable in both homogenous and heterogeneous environments. For example, the techniques described herein can be used to copy and restore data from and to virtual machines 110 operating solely on VMware virtual machine hosts (e.g., VMware ESX servers) or on solely Microsoft virtual machine hosts (e.g., on a Microsoft Virtual Server or a Microsoft Windows Server Hyper-V). As another example, the techniques described herein can be used to copy and restore data from and to virtual machines 110 that are operating in a mixed-vendor environment (e.g., virtual machines from VMware, Microsoft, and/or other vendors). The data agent 155 can perform file-level, volume-level, and/or disk-level copies of virtual machines 110 operating on these Microsoft platforms, and perform restores out of file-level, volume-level and disk-level copies.

For example, virtual machines 110 operating on these Microsoft platforms have their virtual disks 140 in *.vhd files. In performing a disk-level copy of a virtual machine 110 operating on a Microsoft platform, the data agent 155 copies the *.vhd files, extracts metadata (e.g., file, volume, disk relationships metadata) from the *.vhd files and stores this metadata. In restoring out of a disk-level copy, the data agent 155 uses the stored metadata to reconstruct the virtual disks 140, volumes and files to allow the data agent 155 to restore files, volumes or entire virtual machines 110. The techniques described herein can also be used to copy and restore data from and to virtual machines 110 operating on virtual machine hosts 105 from other vendors.

Conversion Between Differing Virtual Machine Formats

In the context of a VMware virtual machine 110, in restoring a volume of a virtual machine 110 (e.g., step 930 of the process 900), the secondary storage computing device 1065 restores the volume as a VMware volume, e.g., to a virtual machine 110 operating on a virtual machine host 105. However, the secondary storage computing device 1065 can also restore the volume as a Microsoft volume, e.g., to a

38

virtual machine 110 operating on Microsoft Virtual Server or Microsoft Windows Server Hyper-V. The secondary storage computing device 1065 can thus convert data in the VMware *.vmdk format to data in the Microsoft *.vhd format. This conversion process can also be performed in the opposite direction, e.g., from the Microsoft *.vhd format to the VMware *.vmdk format.

Similarly, in restoring an entire virtual machine 110 (e.g., step 965 of the process 900), the secondary storage computing device 1065 can restore the entire virtual machine 110 as a virtual machine 110 operating on a Microsoft platform. The secondary storage computing device 1065 does so by converting the data in the *.vmdk format to data in the *.vhd format (and associated configuration files). The secondary storage computing device 1065 can thus convert a virtual machine 110 operating on an ESX Server to a virtual machine 110 operating on Microsoft Virtual Server or Microsoft Windows Server Hyper-V. This conversion process can also be performed in the opposite direction, e.g., from the Microsoft *.vhd format to the VMware *.vmdk format. The conversion process enables virtual machine data originating on VMware platforms to be migrated to other platforms, and for virtual machine data originating on non-VMware platforms to be migrated to the VMware platform. Similar conversions can also be performed for virtual disks 140.

To perform the conversion, the secondary storage computing device 1065 may use APIs or other programmatic techniques. For example, to convert a *.vhd file to a *.vmdk file, the secondary storage computing device 1065 may create the *.vmdk file, create necessary data structures (e.g., grain directories and grain tables) within the *.vmdk file, and copy sectors of the volume of the *.vhd file to the *.vmdk file, going extent by extent and creating necessary entries in the data structures (e.g., entries in the grain directories and grain tables) along the way. The secondary storage computing device 1065 may perform a similar process to convert a *.vmdk file to a *.vhd file. As another example, the secondary storage computing device 1065 may analyze a *.vmdk file using an API function, determine its sectors using another API function, and copy each sector of to a *.vhd file using a third API function. As another example, the secondary storage computing device 1065 may analyze a *.vhd file using an API function, determine its sectors using another API function, and copy each sector of to a *.vmdk file using a third API function. The secondary storage computing device 1065 may use other techniques (e.g., third-party toolkits) to perform conversions between *.vmdk and *.vhd formats.

Conversion between other formats is also possible. For example, the secondary storage computing device 1065 can convert data between the VMware format and an Open Virtual Machine Format (OVF) and vice-versa. Those of skill in the art will understand that a wide variety of conversions are possible, and the techniques are not limited to the conversions described herein.

Secondary Storage Computing Device Index

As described herein, a secondary storage computing device may maintain an index, a database, or other data structure that it uses to store index data generated during storage operations. The secondary storage computing device may use this index data to quickly and efficiently locate data that has been previously copied. This index data may be used for various purposes, such as for browsing by an administrator and/or for restoring the previously copied data.

During a storage operation involving multiple virtual machines 110, the secondary storage computing device

US 9,740,723 B2

39

populates one index with metadata corresponding to all the multiple virtual machines 110 (e.g., a master index). For each of the virtual machines 110, the secondary storage computing device also populates an index with metadata corresponding to that virtual machine 110 (e.g., a sub-index). The master index points to (or refers to) the sub-indices. When an operation to restore virtual machine data is to be performed, the master index is accessed. Because the master index points to the sub-indices, these can be accessed, and the indexed data is used so as to present the virtual machine data that is available to be restored. This available virtual machine data is displayed to an administrator segregated by individual virtual machines 110, which is a logical distinction that is likely intuitive to the administrator. Accordingly, accessing individual virtual machine index data involves two levels of indirection, one for the master index, and one for the sub-indices.

Additionally or alternatively, the secondary storage computing device can populate a single index that is subdivided or otherwise logically separated into multiple sub-indices, one sub-index for each virtual machine 110. When an operation to restore virtual machine data is to be performed, the index data populated by the secondary storage computing device can be used to present the virtual machine data segregated by individual virtual machines 110. Other logical separations and/or segregations of virtual machine data (e.g., by file type, by owner, etc.) are of course possible. Automatic Throttling of Storage Operations

As described herein, a virtual machine host 105 may host multiple virtual machines 110. If a data agent 155 is to perform simultaneous storage operations on a large number of the virtual machines 110, their performance, individually or collectively, may be adversely affected. This potential adverse effect may be attributable to one or more reasons, such as, for example, the snapshotting of virtual machines 110 prior to copying their data (see FIG. 7). There may not necessarily be a linear relationship between the number of storage operations that the data agent 155 performs (or the number of virtual machines 110 upon which the data agent 155 is performing storage operations) and the reduction in performance. For example, performance may decrease linearly with regards to a first number of concurrent storage operations (e.g., ten concurrent storage operations), and then may drastically decrease after surpassing that first number.

Accordingly, it would be beneficial to be able to limit the number of concurrent storage operations being performed upon the virtual machines 110 hosted by a virtual machine host 105. This could be done in one of several ways. First, there could be a hard limit, or threshold, on the number of simultaneous storage operations performed. For example, the data agent 155 could be limited to performing ten simultaneous storage operations (e.g., upon ten different virtual machines 110). The data agent 155 could distribute the ten simultaneous storage operations across the sub-clients corresponding to the virtual machines 110. For example, if a single virtual machine host 105 hosts 50 virtual machines 110 distributed across five sub-clients, the data agent 155 could be limited to performing two simultaneous storage operations (e.g., upon two virtual machines 110) per sub-client.

Second, the number of concurrent storage operations could be limited based upon the performance of one or more individual virtual machines 110 and/or the performance of the virtual machine host 105. The data agent 155 can measure performance using standard metrics (e.g., number of disk writes and/or reads per second, central processing unit (CPU) usage, memory usage, etc.). If the data agent 155

40

determines that the performances of the virtual machines 110 are below a certain performance threshold, the data agent 155 could reduce the number of simultaneous storage operations that it performs. Alternatively, if the data agent 155 determines that the performances of the virtual machines 110 exceed the certain performance threshold, the data agent 155 could increase the number of simultaneous storage operations that it performs.

Third, the throughput of concurrent storage operations could be reduced so as to utilize less of the resources (e.g., CPU, disk, memory, network bandwidth, etc.) of the virtual machines 110 and/or the virtual machine host 105. This reduction in throughput may lessen the loads placed upon the virtual machines 110 and/or the virtual machine host 105 by the simultaneous storage operations. However, this may also necessitate lengthening the window of time in which the storage operations are performed. In each of these three approaches, if the data agent 155 is unable to perform a storage operation upon a virtual machine 110, the data agent 155 may flag the virtual machine 110 for later performance of a storage operation and move to the next virtual machine 110. These three approaches are not mutually exclusive, and combinations of two or more of the three may be used so as to optimally perform storage operations upon virtual machines 110.

Additional Interfaces for Configuring Storage Operations for Virtual Machine Data

FIG. 11 is a display diagram illustrating an example interface 1100 provided by aspects of the invention. The interface 1100 enables an administrator to browse copied virtual machine data for purposes of restoring it. The administrator can specify that the latest data is to be browsed or specify a point in time before which the data is to be browsed using options 1105. The administrator can also select a virtual machine storage manager 145 using list box 1110 and a secondary storage computing device 1065 using list box 1115. The administrator can also select the intended type of restore using options 1120: either restoration of individual files and/or folders, restoration of entire volumes, or restoration of virtual machines and/or virtual disks.

FIG. 12 is a display diagram illustrating example interfaces 1200 and 1250 provided by aspects of the invention. The interface 1200 may be shown after the administrator has selected to browse the latest data (e.g., reference character 1105 of FIG. 11) and the selected intended restoration is that of individual files and/or folders (e.g., reference character 1120 of FIG. 11). The interface 1200 includes a folder structure 1205 corresponding to the copied virtual machine data. As shown, a folder 1208 within a volume (Volume 1) of a virtual machine (TESTVM111) is selected. The interface 1250 provides the administrator with options for restoring the selected folder. These include an option 1210 to restore ACLs associated with the virtual machine data and an option 1215 to unconditionally overwrite data. The administrator can specify the destination computer and folder in region 1220. The administrator can also specify options for preserving or removing source paths in region 1225.

FIGS. 13A and 13B are display diagrams illustrating example interfaces 1300 and 1340 provided by aspects of the invention. The interface 1300 may be shown after the administrator has selected the intended restoration to be that of an entire volume (e.g., reference character 1120 of FIG. 11). The interface 1300 allows the administrator to select to restore a volume as a physical volume, as a *.vhd file (corresponding to Microsoft virtual machines), or as a *.vmdk file (corresponding to VMware virtual machines)

US 9,740,723 B2

41

using options **1305**. The administrator can also select a destination computer in list box **1310**, a source volume to be restored in region **1315**, and a destination volume using button **1320**. Selecting the button **1320** causes the interface **1340** to be displayed, which allows the administrator to select a mount point on the selected destination computer from available mount points listed in region **1325**.

FIG. **13B** illustrates the interface **1300** when the administrator has selected to restore a volume as a *.vhd file from the options **1305**. The administrator can select a destination computer in list box **1310** and a destination folder for the *.vhd file can be selected using button **1335**. Once selected, the destination folder will be displayed in text box **1330**.

FIGS. **14A** and **14B** are display diagrams illustrating an example interface **1400** provided by aspects of the invention. The interface **1400** may be shown after the administrator has selected the intended restoration to be that of virtual machines or virtual disks (e.g., reference character **1120** of FIG. **11**). The interface **1400** allows the administrator to select to restore either a virtual machine or virtual disks. As with the interface **1300**, the administrator can select a destination computer in list box **1410** and a destination folder using button **1435**. Once selected, the destination folder will be displayed in text box **1430**. If restore of virtual machines is selected (FIG. **14A**), the administrator can provide the name of the virtual machine to be restored in text box **1415**, and the name of the server to which it is to be restored in text box **1420**. If the virtual machine is to be restored to a virtual machine host **105**, the administrator selects this option **1425** and specifies the name of the virtual machine host **105** in text box **1420**. If the virtual machine is to be restored to a virtual machine host managed by a virtual machine manager **202**, the administrator selects this option **1425** and provides the name of the virtual machine manager **202** in text box **1420** and the name of the virtual machine host **105** in text box **1440**. The administrator also specifies authentication credentials in region **1445**.

FIG. **15** is a display diagram illustrating an example interface **1500** provided by aspects of the invention. The interface **1500** allows the administrator to specify options for storage operations for a sub-client. Region **1505** displays information associated with the sub-client. The administrator can specify the number of data readers to use in performing storage operations using spinner **1510**. The specified number of data readers corresponds to the number of storage operations to be simultaneous performed on the virtual machines **110** associated with the sub-client. As described herein, the number of simultaneous storage operations may be limited or capped so as not to adversely affect performance of the virtual machines **110**.

The administrator can also specify the type of copy operation to be performed using options **1515**: either file level, volume level, or disk level. The administrator can also select one or more virtual machine storage managers **145** that are to perform the copy operations using list box **1520**. Generally, the administrator has to select at least one virtual machine storage manager **145** to perform the copy operation.

If the administrator selects two or more virtual machine storage managers **145** in the list box **1520**, this causes the copy operation, when it commences, to be performed by the selected virtual machine storage managers **145**. This can assist in load balancing and provide other benefits. For example, one or more sub-clients could be configured to perform copy operations upon all the virtual machines **110** associated with a specific virtual machine manager **202**. This could be a large number of virtual machines **110**, and if only

42

one virtual machine storage manager **145** were to perform copy operations upon the one or more sub-clients' virtual machines **110**, it could take a lengthy period of time to conclude all the copy operations. Accordingly, distributing copy operations across multiple virtual machine storage managers **145** can shorten the amount of time it takes to conclude all the copy operations. This can be true even in the case of a single virtual machine **110** (for example, when the single virtual machine **110** contains a large amount of data). This workload balancing can provide significant benefits, such as when copy operations need to be performed entirely within a specific window of time (e.g., from 2:00 a.m. to 4:00 a.m.). Moreover, such load balancing only requires a single virtual machine storage manager **145** to coordinate the performance of the copy operations by the multiple virtual machine storage managers **145**.

For example, an administrator could select a first virtual machine storage manager **145** that coordinates the copying of data of multiple virtual machines **110**. The administrator could also select one or more second virtual machine storage managers **145** to perform the copying of data of multiple virtual machines **110**. The first data agent **155** can allocate responsibility for the copying of the data amongst the second virtual machine storage managers **145** such that the copying is more or less evenly distributed based upon selections previously made (static load-balancing).

Additionally or alternatively, the first virtual machine storage manager **145** can distribute the copy operations across the second virtual machine storage managers **145** based upon various factors. Consider an example where ten copy operations of the data of ten virtual machines **110** are to be performed, and where two second virtual machine storage managers **145** can be used to perform the copy operations. The first virtual machine storage manager **145** can determine an availability of the second virtual machine storage managers **145**, as measured by percentage of CPU usage, percentage of network utilization, disk utilization, average time spent performing storage operations, and/or other factors. For example, if the first virtual machine storage manager **145** determines that one of the second virtual machine storage managers **145** have a percentage of CPU usage of 10%, and that the other second virtual machine storage manager **145** has a percentage of CPU usage of 50%, the storage manager **1005** may allocate eight of the copy operations to the one second virtual machine storage manager **145** and the remaining two copy operations to the other second virtual machine storage manager **145**, based upon this measurement of availability (dynamic load-balancing). The first virtual machine storage manager **145** may also use other factors known to those of skill in the art to balance the workloads of the two virtual machine storage managers **145**. Additionally or alternatively, the storage manager **1005** may perform the load-balancing amongst the multiple virtual machine storage managers **145**.

Copying of Virtual Machine Data on an Incremental Basis

As described herein, the primary storage data store **135** stores the data of virtual machines **110**. The data is organized into multiple blocks of fixed size (e.g., 64 kb, 128 kb, 256 kb, 512 kb, etc.). A data agent **155** can perform full copies of data of virtual machines **110** using the blocks of data. In some instances, it may not be necessary to perform a second full backup of virtual machine data after a first full backup has been performed (at least not until a set period of time has elapsed). Rather, incremental and/or differential backups of virtual machine data may suffice.

FIG. **16** is a flow diagram illustrating a process **1600** for copying virtual machine data on an incremental basis (or a

US 9,740,723 B2

43

differential basis, but incremental copies are described herein for brevity). The process 1600 may be performed by the data agent 155. The data agent 155 begins at step 1605 by accessing data structures within virtual disk files 140. As described herein, virtual disks 140 can be growable or preallocated. In either case, virtual disks 140 may use internal data structures to specify the blocks that have been allocated and/or are being used by the virtual machines 110. For example, VMware virtual machine disk files (*.vmdk files) include grain directories and grain tables, and Microsoft virtual disk files (*.vhd files) include block allocation tables. These internal data structures specify the blocks within virtual disks 140 that have been allocated and/or are being used for data storage.

At step 1610, the data agent 155 determines the blocks that have been allocated and/or are being used within the virtual disks 140. At step 1615 the data agent 155 accesses a block identifier data structure to make the determination of which blocks have changed since the last storage operation involving a full copy of the virtual machine data.

FIG. 17 is a diagram illustrating an example table 1700 that may be employed as a block identifier data structure. The data agent 155 may create the table 1700 during, for example, a storage operation that performs a full copy of all of the data of the virtual machine 110. The table 1700 includes a block identifier column 1702 and a substantially unique identifier column 1704. The block identifier column 1702 stores identifiers of blocks within a virtual disk 140. Block may be identified by their order within a virtual disk 140. For example, a first block may have an identifier of one ("1"), a second block may have an identifier of two ("2"), and so forth. The substantially unique identifier column 1704 stores identifiers generated for the block by the data agent 155. For example, substantially unique identifiers could be generated using Message Digest Algorithm 5 (MD5) or Secure Hash Algorithm SHA 512. Although the table 1700 is illustrated as including three rows 1706 of three different blocks, the table 1700 generally includes one row for each block in a virtual disk 140.

Returning to FIG. 16, at step 1620, for each block that the data agent 155 determines has been allocated and/or is in use, the data agent 155 generates a substantially unique identifier. At step 1625, the data agent 155 finds the row in the table 1700 for which the block identifier of column 1702 is the same as the block identifier of the block currently being processed. The data agent 155 then looks up the substantially unique identifier in the column 1704, and compares it to the generated substantially unique identifier. If the two substantially unique identifiers do not match, then the block currently being processed has changed. The process 1600 then continues at step 1630 where the data agent 155 copies the block to a storage device. The data agent 155 then updates the column 1704 of the table 1700 with the generated substantially unique identifier. At step 1640, the data agent 155 determines whether there are more blocks to process. If so, the process 1600 returns to step 1620. If not, the process 1600 concludes. If the block has not changed (step 1625), the process 1600 continues at step 1640. The next time the data agent 155 performs a full copy of all of the data of the virtual machine 110, the data agent 155 can regenerate substantially unique identifiers for blocks of data and repopulate or recreate the table 1700.

If, at step 1625, the data agent 155 cannot find a row in the table 1700 for which the block identifier of column 1702 is the same as the block identifier of the block currently being processed, this generally indicates that the data agent 155 is currently processing a block that has been allocated

44

and/or has been put to use since the time at which the last full copy operation was performed. If this is the case, the data agent 155 will copy the block to the storage device, and at step 1635 the data agent will add a row to the table 1700 with the block identifier and the generated substantially unique identifier.

The process 1600 and the table 1700 thus enable copying of virtual machine data on an incremental basis. This can provide significant advantages in that it allows for only copying the data that has changed while still providing for protection of virtual machine data. Changes can be made to the process 1600 and/or the table 1700 while still retaining the ability to perform storage operations on an incremental basis. For example, a monitoring agent could monitor the blocks of the virtual disks 140 and, each time a block is changed (e.g., due to a write operation), the monitoring agent could set a flag (or bit) for the block in a data structure. When the data agent 155 is to perform an incremental copy, it can access the data structure containing the flags and only copy blocks that have been flagged. As another example, the table 1700 could include a time copied column to store timestamps of when a block was last copied to a storage device. If the difference between the time of the incremental copy operation and the last time copied is greater than a threshold time, the data agent 155 could copy the block to the storage device, regardless of whether the generated substantially unique identifier matches the stored substantially unique identifier.

CONCLUSION

From the foregoing, it will be appreciated that specific embodiments of the storage system have been described herein for purposes of illustration, but that various modifications may be made without deviating from the spirit and scope of the invention. For example, although copy operations have been described, the system may be used to perform many types of storage operations (e.g., backup operations, restore operations, archival operations, copy operations, CDR operations, recovery operations, migration operations, HSM operations, etc.). Accordingly, the invention is not limited except as by the appended claims.

Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense; that is to say, in the sense of "including, but not limited to." The word "coupled," as generally used herein, refers to two or more elements that may be either directly connected, or connected by way of one or more intermediate elements. Additionally, the words "herein," "above," "below," and words of similar import, when used in this application, shall refer to this application as a whole and not to any particular portions of this application. Where the context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number respectively. The word "or" in reference to a list of two or more items, that word covers all of the following interpretations of the word: any of the items in the list, all of the items in the list, and any combination of the items in the list.

The above detailed description of embodiments of the invention is not intended to be exhaustive or to limit the invention to the precise form disclosed above. While specific embodiments of, and examples for, the invention are described above for illustrative purposes, various equivalent modifications are possible within the scope of the invention,

US 9,740,723 B2

45

as those skilled in the relevant art will recognize. For example, while processes or blocks are presented in a given order, alternative embodiments may perform routines having steps, or employ systems having blocks, in a different order, and some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified. Each of these processes or blocks may be implemented in a variety of different ways. Also, while processes or blocks are at times shown as being performed in series, these processes or blocks may instead be performed in parallel, or may be performed at different times.

The teachings of the invention provided herein can be applied to other systems, not necessarily the system described above. The elements and acts of the various embodiments described above can be combined to provide further embodiments.

These and other changes can be made to the invention in light of the above Detailed Description. While the above description details certain embodiments of the invention and describes the best mode contemplated, no matter how detailed the above appears in text, the invention can be practiced in many ways. Details of the system may vary considerably in implementation details, while still being encompassed by the invention disclosed herein. As noted above, particular terminology used when describing certain features or aspects of the invention should not be taken to imply that the terminology is being redefined herein to be restricted to any specific characteristics, features, or aspects of the invention with which that terminology is associated. In general, the terms used in the following claims should not be construed to limit the invention to the specific embodiments disclosed in the specification, unless the above Detailed Description section explicitly defines such terms. Accordingly, the actual scope of the invention encompasses not only the disclosed embodiments, but also all equivalent ways of practicing or implementing the invention under the claims.

While certain aspects of the invention are presented below in certain claim forms, the inventors contemplate the various aspects of the invention in any number of claim forms. For example, while only one aspect of the invention is recited as embodied in a computer-readable medium, other aspects may likewise be embodied in a computer-readable medium. As another example, while only one aspect of the invention is recited as a means-plus-function claim under 35 U.S.C. §112, sixth paragraph, other aspects may likewise be embodied as a means-plus-function claim, or in other forms, such as being embodied in a computer-readable medium. (Any claims intended to be treated under 35 U.S.C. §112, ¶6 will begin with the words “means for.”) Accordingly, the inventors reserve the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the invention.

We claim:

1. An automated method of managing data of one or more virtual machines managed by a virtual machine manager, wherein the method is performed by one or more physical computing systems, each computing system having a processor and memory, the method comprising:

receiving a request to copy data of one or more virtual machines to a physical storage device;

determining whether the one or more virtual machines are managed by a virtual machine manager,

wherein the determining is performed by reading a stored indication of the virtual machine manager, or by scanning a network for the virtual machine manager;

46

based on the determining that the one or more virtual machines are managed by the virtual machine manager, automatically accessing the virtual machine manager; automatically requesting information from the virtual machine manager regarding virtual machines that the virtual machine manager manages;

automatically determining, by the one or more computing systems, from information provided by the virtual machine manager, the virtual machines managed by the virtual machine manager; and

for at least one of the determined virtual machines, copying the data of the virtual machine to the physical storage device.

2. The automated method of claim 1, wherein the request further specifies at least one of the following:

that a file-level copy is to be performed for at least one virtual machine managed by the virtual machine manager;

that a volume-level copy is to be performed for at least one virtual machine managed by the virtual machine manager; and

that a disk-level copy is to be performed for at least one virtual machine managed by the virtual machine manager.

3. The automated method of claim 1, wherein the request specifies that a disk-level copy is to be performed for at least one virtual machine managed by the virtual machine manager, and wherein the method further comprises:

determining a virtual machine disk file in which the at least one virtual machine stores data;

copying the virtual machine disk file from a filesystem of a virtual machine host hosting the at least one virtual machine to the storage device;

extracting information from the virtual machine disk file, wherein extracting information includes extracting information that describes a location of a virtual file allocation table utilized by the at least one virtual machine;

associating the extracted information with the at least one virtual machine; and

storing the extracted information in association with the at least one virtual machine.

4. The automated method of claim 1, wherein the request specifies that a disk-level copy is to be performed for at least one virtual machine managed by the virtual machine manager, and wherein the method further comprises:

determining at least two virtual machine disk files in which the at least one virtual machine stores data;

copying the at least two virtual machine disk files from a filesystem of a virtual machine host hosting the at least one virtual machine to the storage device;

extracting from the at least two virtual machine disk files information that describes the relationships between the virtual machine disk files;

extracting from the at least two virtual machine disk files information that describes at least one virtual volume contained within one or more of the at least two virtual machine disk files;

extracting from the at least two virtual machine disk files information that describes at least one virtual file allocation table that describes locations of virtual files contained within the at least one virtual volume; and storing the extracted information.

US 9,740,723 B2

47

5. The automated method of claim 1, further comprising, for at least one virtual machine:

determining distinct data objects within the virtual machine data;

indexing the distinct data objects; and

storing the indexed distinct data objects in an index, wherein the index also includes indexed distinct data objects associated with at least one non-virtual machine.

6. The automated method of claim 1, further comprising: determining distinct data objects within the virtual machine data;

indexing the distinct data objects within the virtual machine data;

storing the indexed distinct data objects in an index, wherein the index also includes indexed distinct data objects associated with at least one non-virtual machine;

receiving a request to query the index; and

returning search results responsive to the request, wherein the search results include search results associated with distinct data objects of the least one virtual machine and with distinct data objects at least one non-virtual machine.

7. The automated method of claim 1, further comprising: determining distinct data objects within the virtual machine data; and

for at least one of the distinct data objects:

generating a substantially unique identifier for the distinct data object;

determining, based on the substantially unique identifier, if an instance of the distinct data object has already been stored on a single instance storage device; and

if an instance of the distinct data object has not already been stored on the single instance storage device, then storing the data object on the single instance storage device.

8. The automated method of claim 1 wherein, for at least one virtual machine, copying the data of the virtual machine to the storage device includes determining a virtual machine disk file in which the virtual machine stores data and copying the virtual machine disk file from a filesystem of a virtual machine host hosting the virtual machine to the storage device, wherein the virtual machine is a first type of virtual machine and the virtual machine disk file is in a first format utilized by the first type of virtual machine, and wherein the method further comprises:

receiving a request to convert the virtual machine disk file to a virtual machine disk file in a second format utilized by a second type of virtual machine;

accessing the virtual machine disk file stored on the storage device;

converting the virtual machine disk file to a virtual machine disk file in the second format; and

copying the converted virtual machine disk file in the second format to a location where it may be utilized by a virtual machine of the second type.

9. The automated method of claim 1 wherein automatically determining includes automatically determining multiple virtual machines hosted by a virtual machine host, and wherein the method further comprises:

determining a threshold number of simultaneous storage operations that can be performed upon the data of the multiple virtual machines hosted by the virtual machine host; and

48

performing multiple storage operations upon the data of the multiple virtual machines hosted by the virtual machine host,

wherein the number of simultaneous copy operations performed upon the data of the multiple virtual machines hosted by the virtual machine host does not exceed the threshold number.

10. The automated method of claim 1, wherein automatically determining includes automatically determining multiple virtual machines, and wherein the method further comprises:

populating a first index with metadata corresponding to data of a first virtual machine, wherein the metadata includes a location of the data of the first virtual machine on the storage device;

populating a second index with metadata corresponding to the multiple virtual machines, wherein the metadata includes a reference to the first index;

receiving a request to access data of the first virtual machine on the storage device;

analyzing the metadata included in the second index;

determining the reference to the first index;

analyzing the metadata included in the first index; and

displaying metadata corresponding to the data of the first virtual machine.

11. At least one computer-readable medium, excluding transitory signals, and storing computer-executable instructions to perform an automated method of managing data of one or more virtual machines managed by a virtual machine manager, wherein the method is performed by one or more physical computing systems, each computing system having a processor and memory, the method comprising:

receiving a request to copy data of one or more virtual machines to a physical storage device;

determining whether the one or more virtual machines are managed by a virtual machine manager,

wherein the determining is performed by reading a stored indication of the virtual machine manager, or by scanning a network for the virtual machine manager;

based on the determining that the one or more virtual machines are managed by the virtual machine manager, automatically accessing the virtual machine manager; automatically requesting information from the virtual machine manager regarding virtual machines that the virtual machine manager manages;

automatically determining, by the one or more computing systems, from information provided by the virtual machine manager, the virtual machines managed by the virtual machine manager; and

for at least one of the determined virtual machines, copying the data of the virtual machine to the physical storage device.

12. The computer-readable medium of claim 11, wherein the request specifies that a disk-level copy is to be performed for at least one virtual machine managed by the virtual machine manager, and wherein the method further comprises:

determining a virtual machine disk file in which the at least one virtual machine stores data;

copying the virtual machine disk file from a filesystem of a virtual machine host hosting the at least one virtual machine to the storage device;

extracting information from the virtual machine disk file, wherein extracting information includes extracting

US 9,740,723 B2

49

information that describes a location of a virtual file allocation table utilized by the at least one virtual machine;
 associating the extracted information with the at least one virtual machine; and
 storing the extracted information in association with the at least one virtual machine.

13. The computer-readable medium of claim 11, wherein the request specifies that a disk-level copy is to be performed for at least one virtual machine managed by the virtual machine manager, and wherein the method further comprises:

determining at least two virtual machine disk files in which the at least one virtual machine stores data;
 copying the at least two virtual machine disk files from a filesystem of a virtual machine host hosting the at least one virtual machine to the storage device;
 extracting from the at least two virtual machine disk files information that describes the relationships between the virtual machine disk files;
 extracting from the at least two virtual machine disk files information that describes at least one virtual volume contained within one or more of the at least two virtual machine disk files;
 extracting from the at least two virtual machine disk files information that describes at least one virtual file allocation table that describes locations of virtual files contained within the at least one virtual volume; and
 storing the extracted information.

14. The computer-readable medium of claim 11, further comprising:

determining distinct data objects within the virtual machine data;
 indexing the distinct data objects within the virtual machine data;
 storing the indexed distinct data objects in an index, wherein the index also includes indexed distinct data objects associated with at least one non-virtual machine;
 receiving a request to query the index; and
 returning search results responsive to the request, wherein the search results include search results associated with distinct data objects of the least one virtual machine and with distinct data objects at least one non-virtual machine.

15. The computer-readable medium of claim 11, further comprising:

determining distinct data objects within the virtual machine data; and
 for at least one of the distinct data objects:
 generating a substantially unique identifier for the distinct data object;
 determining, based on the substantially unique identifier, if an instance of the distinct data object has already been stored on a single instance storage device; and
 if an instance of the distinct data object has not already been stored on the single instance storage device, then storing the data object on the single instance storage device.

16. A system configured to manage data of one or more virtual machines in a computing network, the system comprising:

at least one physical processor;
 at least one physical memory device, coupled to the at least one processor, and storing instructions to be

50

executed by the at least one processor to perform a method, the method comprising:

receiving a request to copy data of one or more virtual machines to a physical storage device;

determining whether the one or more virtual machines are managed by a virtual machine manager, wherein the determining is performed by reading a stored indication of the virtual machine manager, or by scanning a network for the virtual machine manager;

based on the determining that the one or more virtual machines are managed by the virtual machine manager, automatically accessing the virtual machine manager;

automatically requesting information from the virtual machine manager regarding virtual machines that the virtual machine manager manages;

automatically determining, by the one or more computing systems, from information provided by the virtual machine manager, the virtual machines managed by the virtual machine manager; and

for at least one of the determined virtual machines, copying the data of the virtual machine to the physical storage device.

17. The system of claim 16, wherein the request specifies that a disk-level copy is to be performed for at least one virtual machine managed by the virtual machine manager, and wherein the method further comprises:

determining a virtual machine disk file in which the at least one virtual machine stores data;

copying the virtual machine disk file from a filesystem of a virtual machine host hosting the at least one virtual machine to the storage device;

extracting information from the virtual machine disk file, wherein extracting information includes extracting information that describes a location of a virtual file allocation table utilized by the at least one virtual machine;

associating the extracted information with the at least one virtual machine; and

storing the extracted information in association with the at least one virtual machine.

18. The system of claim 16, wherein the request specifies that a disk-level copy is to be performed for at least one virtual machine managed by the virtual machine manager, and wherein the method further comprises:

determining at least two virtual machine disk files in which the at least one virtual machine stores data;

copying the at least two virtual machine disk files from a filesystem of a virtual machine host hosting the at least one virtual machine to the storage device;

extracting from the at least two virtual machine disk files information that describes the relationships between the virtual machine disk files;

extracting from the at least two virtual machine disk files information that describes at least one virtual volume contained within one or more of the at least two virtual machine disk files;

extracting from the at least two virtual machine disk files information that describes at least one virtual file allocation table that describes locations of virtual files contained within the at least one virtual volume; and
 storing the extracted information.

19. The system of claim 16, wherein the method further comprises:

determining distinct data objects within the virtual machine data;

US 9,740,723 B2

51

indexing the distinct data objects within the virtual machine data;
storing the indexed distinct data objects in an index, wherein the index also includes indexed distinct data objects associated with at least one non-virtual machine;
receiving a request to query the index; and
returning search results responsive to the request, wherein the search results include search results associated with distinct data objects of the least one virtual machine and with distinct data objects at least one non-virtual machine.
20. The system of claim 16, wherein the method further comprises:
determining distinct data objects within the virtual machine data; and
for at least one of the distinct data objects:
generating a substantially unique identifier for the distinct data object;

52

determining, based on the substantially unique identifier, if an instance of the distinct data object has already been stored on a single instance storage device; and
if an instance of the distinct data object has not already been stored on the single instance storage device, then storing the data object on the single instance storage device.
21. The system of claim 16, wherein the request further specifies at least one of the following:
that a file-level copy is to be performed for at least one virtual machine managed by the virtual machine manager;
that a volume-level copy is to be performed for at least one virtual machine managed by the virtual machine manager; and
that a disk-level copy is to be performed for at least one virtual machine managed by the virtual machine manager.

* * * * *

EXHIBIT E

(12) **United States Patent**
Sancheti

(10) **Patent No.:** **US 10,210,048 B2**
(45) **Date of Patent:** **Feb. 19, 2019**

(54) **SELECTIVE SNAPSHOT AND BACKUP COPY OPERATIONS FOR INDIVIDUAL VIRTUAL MACHINES IN A SHARED STORAGE**

(71) Applicant: **Commvault Systems, Inc.**, Tinton Falls, NJ (US)

(72) Inventor: **Ashwin Gautamchand Sancheti**, Ocean, NJ (US)

(73) Assignee: **Commvault Systems, Inc.**, Tinton Falls, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 73 days.

(21) Appl. No.: **15/334,127**

(22) Filed: **Oct. 25, 2016**

(65) **Prior Publication Data**

US 2018/0113623 A1 Apr. 26, 2018

(51) **Int. Cl.**
G06F 12/00 (2006.01)
G06F 11/14 (2006.01)
G06F 9/455 (2018.01)

(52) **U.S. Cl.**
CPC **G06F 11/1451** (2013.01); **G06F 9/45558** (2013.01); **G06F 2009/45575** (2013.01); **G06F 2201/84** (2013.01)

(58) **Field of Classification Search**
CPC G06F 3/065; G06F 3/0664; G06F 3/0619; G06F 3/0673

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,084,231 A 4/1978 Capozzi et al.
4,267,568 A 5/1981 Dechant et al.
4,283,787 A 8/1981 Chambers
(Continued)

FOREIGN PATENT DOCUMENTS

EP 0259912 3/1988
EP 0405926 1/1991
(Continued)

OTHER PUBLICATIONS

Arneson, "Mass Storage Archiving in Network Environments" IEEE, Oct. 31-Nov. 1998, pp. 45-50.

(Continued)

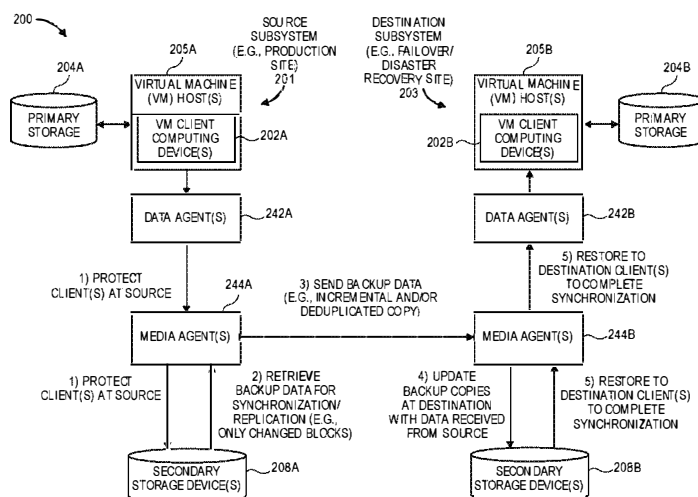
Primary Examiner — Gurtej Bansal

(74) *Attorney, Agent, or Firm* — Commvault Systems, Inc.

(57) **ABSTRACT**

System and techniques for performing selective snapshot and backup copy operations for individual virtual machines in a shared storage. The system can include a hypervisor configured to create and operate a plurality of virtual machines. The system can also include one or more shared physical computer storage devices communicatively coupled to the hypervisor to store the plurality of virtual machines. A plurality of storage volumes can be provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines. The system can also include a virtual server agent configured to issue a command to the hypervisor to perform a snapshot or backup copy operation for a selected one of the plurality of virtual machines without performing the operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

20 Claims, 13 Drawing Sheets



US 10,210,048 B2

Page 2

(56)

References Cited

U.S. PATENT DOCUMENTS

4,417,321 A 11/1983 Chang et al.
 4,641,274 A 2/1987 Swank
 4,654,819 A 3/1987 Stiffler et al.
 4,686,620 A 8/1987 Ng
 4,912,637 A 3/1990 Sheedy et al.
 4,995,035 A 2/1991 Cole et al.
 5,005,122 A 4/1991 Griffin et al.
 5,093,912 A 3/1992 Dong et al.
 5,133,065 A 7/1992 Cheffetz et al.
 5,193,154 A 3/1993 Kitajima et al.
 5,212,772 A 5/1993 Masters
 5,226,157 A 7/1993 Nakano et al.
 5,239,647 A 8/1993 Anglin et al.
 5,241,668 A 8/1993 Eastridge et al.
 5,241,670 A 8/1993 Eastridge et al.
 5,276,860 A 1/1994 Fortier et al.
 5,276,867 A 1/1994 Kenley et al.
 5,287,500 A 2/1994 Stoppani, Jr.
 5,301,286 A 4/1994 Rajani
 5,321,816 A 6/1994 Rogan et al.
 5,347,653 A 9/1994 Flynn et al.
 5,410,700 A 4/1995 Fecteau et al.
 5,420,996 A 5/1995 Aoyagi
 5,454,099 A 9/1995 Myers et al.
 5,559,991 A 9/1996 Kanfi
 5,642,496 A 6/1997 Kanfi
 6,418,478 B1 7/2002 Ignatius et al.
 6,542,972 B2 4/2003 Ignatius et al.
 6,557,089 B1 4/2003 Reed et al.
 6,658,436 B2 12/2003 Shinsky et al.
 6,721,767 B2 4/2004 De Meno et al.
 6,760,723 B2 7/2004 Shinsky et al.
 7,003,641 B2 2/2006 Prahlad et al.
 7,035,880 B1 4/2006 Crescenti et al.
 7,107,298 B2 9/2006 Prahlad et al.
 7,130,970 B2 10/2006 Devassy et al.
 7,162,496 B2 1/2007 Amarendran et al.
 7,174,433 B2 2/2007 Kottomtharayil et al.
 7,246,207 B2 7/2007 Kottomtharayil et al.
 7,251,713 B1 7/2007 Zhang
 7,315,923 B2 1/2008 Retnamma et al.
 7,343,453 B2 3/2008 Prahlad et al.
 7,389,311 B1 6/2008 Crescenti et al.
 7,395,282 B1 7/2008 Crescenti et al.
 7,440,982 B2 10/2008 Lu et al.
 7,454,569 B2 11/2008 Kavuri et al.
 7,490,207 B2 2/2009 Amarendran et al.
 7,500,053 B1 3/2009 Kavuri et al.
 7,529,782 B2 5/2009 Prahlad et al.
 7,536,291 B1 5/2009 Vijayan Retnamma et al.
 7,543,125 B2 6/2009 Gokhale
 7,546,324 B2 6/2009 Prahlad et al.
 7,603,386 B2 10/2009 Amarendran et al.
 7,606,844 B2 10/2009 Kottomtharayil
 7,613,752 B2 11/2009 Prahlad et al.
 7,617,253 B2 11/2009 Prahlad et al.
 7,617,262 B2 11/2009 Prahlad et al.
 7,620,710 B2 11/2009 Kottomtharayil et al.
 7,636,743 B2 12/2009 Erofeev
 7,651,593 B2 1/2010 Prahlad et al.
 7,657,550 B2 2/2010 Prahlad et al.
 7,660,807 B2 2/2010 Prahlad et al.
 7,661,028 B2 2/2010 Erofeev
 7,734,669 B2 6/2010 Kottomtharayil et al.
 7,747,579 B2 6/2010 Prahlad et al.
 7,801,864 B2 9/2010 Prahlad et al.
 7,809,914 B2 10/2010 Kottomtharayil et al.
 8,156,086 B2 4/2012 Lu et al.
 8,170,995 B2 5/2012 Prahlad et al.
 8,205,049 B1 6/2012 Armangau
 8,229,954 B2 7/2012 Kottomtharayil et al.
 8,230,195 B2 7/2012 Amarendran et al.
 8,245,128 B1 8/2012 Ahad
 8,285,681 B2 10/2012 Prahlad et al.
 8,307,177 B2 11/2012 Prahlad et al.

8,326,803 B1 12/2012 Stringham
 8,346,727 B1 1/2013 Chester
 8,364,652 B2 1/2013 Vijayan et al.
 8,370,542 B2 2/2013 Lu et al.
 8,577,845 B2 11/2013 Nguyen et al.
 8,578,120 B2 11/2013 Attarde et al.
 8,656,123 B2 2/2014 Lee
 8,719,226 B1 5/2014 Jiang
 8,789,208 B1 7/2014 Sundaram et al.
 8,954,446 B2 2/2015 Vijayan et al.
 9,020,900 B2 4/2015 Vijayan et al.
 9,098,495 B2 8/2015 Gokhale
 9,239,687 B2 1/2016 Vijayan et al.
 9,239,840 B1 1/2016 Acharya
 9,286,110 B2 3/2016 Mitkar et al.
 9,298,715 B2 3/2016 Kumarasamy et al.
 9,311,121 B2 4/2016 Deshpande et al.
 9,342,537 B2 5/2016 Kumarasamy et al.
 9,448,731 B2 5/2016 Nallathambi et al.
 9,471,441 B1* 10/2016 Lyadvinsky G06F 11/1464
 9,471,578 B2 10/2016 Nallathambi et al.
 9,495,251 B2 11/2016 Kottomtharayil et al.
 9,495,404 B2 11/2016 Kumarasamy et al.
 9,535,907 B1 1/2017 Stringham
 9,639,426 B2 5/2017 Pawar et al.
 9,703,584 B2 7/2017 Kottomtharayil et al.
 9,710,465 B2 7/2017 Dornemann et al.
 9,740,702 B2 8/2017 Pawar et al.
 9,774,672 B2 9/2017 Nallathambi et al.
 9,886,346 B2 2/2018 Kumarasamy et al.
 2004/0143642 A1 7/2004 Beckmann et al.
 2004/0148376 A1 7/2004 Rangan et al.
 2005/0033878 A1 2/2005 Pangal et al.
 2005/0060598 A1 3/2005 Klotz
 2005/0066118 A1 3/2005 Perry
 2005/0066225 A1 3/2005 Rowan
 2006/0136771 A1 6/2006 Watanabe
 2006/0224846 A1 10/2006 Amarendran et al.
 2007/0115738 A1 5/2007 Emaru
 2007/0185938 A1 8/2007 Prahlad
 2007/0283111 A1 12/2007 Berkowitz et al.
 2009/0319534 A1 12/2009 Gokhale
 2010/0011178 A1 1/2010 Feathergill
 2010/0036931 A1 2/2010 Certain et al.
 2010/0070726 A1* 3/2010 Ngo G06F 11/1469
 711/162
 2010/0122053 A1 5/2010 Prahlad
 2011/0047195 A1 2/2011 Le
 2011/0047340 A1 2/2011 Olson et al.
 2011/0107025 A1* 5/2011 Urkude G06F 11/2094
 711/112
 2011/0295806 A1 12/2011 Erofeev
 2012/0079221 A1 3/2012 Sivasubramanian
 2012/0150818 A1 6/2012 Vijayan Retnamma et al.
 2012/0150826 A1 6/2012 Vijayan Retnamma et al.
 2013/0007183 A1 1/2013 Sorenson et al.
 2014/0201157 A1 7/2014 Pawar et al.
 2014/0201170 A1 7/2014 Vijayan et al.
 2015/0074536 A1 3/2015 Varadharajan et al.
 2016/0019317 A1 1/2016 Pawar et al.
 2016/0132400 A1 5/2016 Pawar et al.
 2016/0139836 A1 5/2016 Nallathambi
 2016/0147607 A1 5/2016 Dornemann et al.
 2016/0350391 A1 12/2016 Vijayan et al.
 2017/0168903 A1 6/2017 Dornemann et al.
 2017/0185488 A1 6/2017 Kumarasamy et al.
 2017/0193003 A1 7/2017 Vijayan et al.
 2017/0235647 A1 8/2017 Kilaru et al.
 2017/0242871 A1 8/2017 Kilaru et al.
 2018/0113622 A1 4/2018 Sancheti

FOREIGN PATENT DOCUMENTS

EP 0467546 1/1992
 EP 0541281 5/1993
 EP 0774715 5/1997
 EP 0809184 11/1997
 EP 0899662 3/1999
 EP 0981090 2/2000

US 10,210,048 B2Page 3

(56)

References Cited

FOREIGN PATENT DOCUMENTS

W●	W● 95/13580	5/1995
W●	W● 99/12098	3/1999
W●	W● 2006/052872	5/2006

OTHER PUBLICATIONS

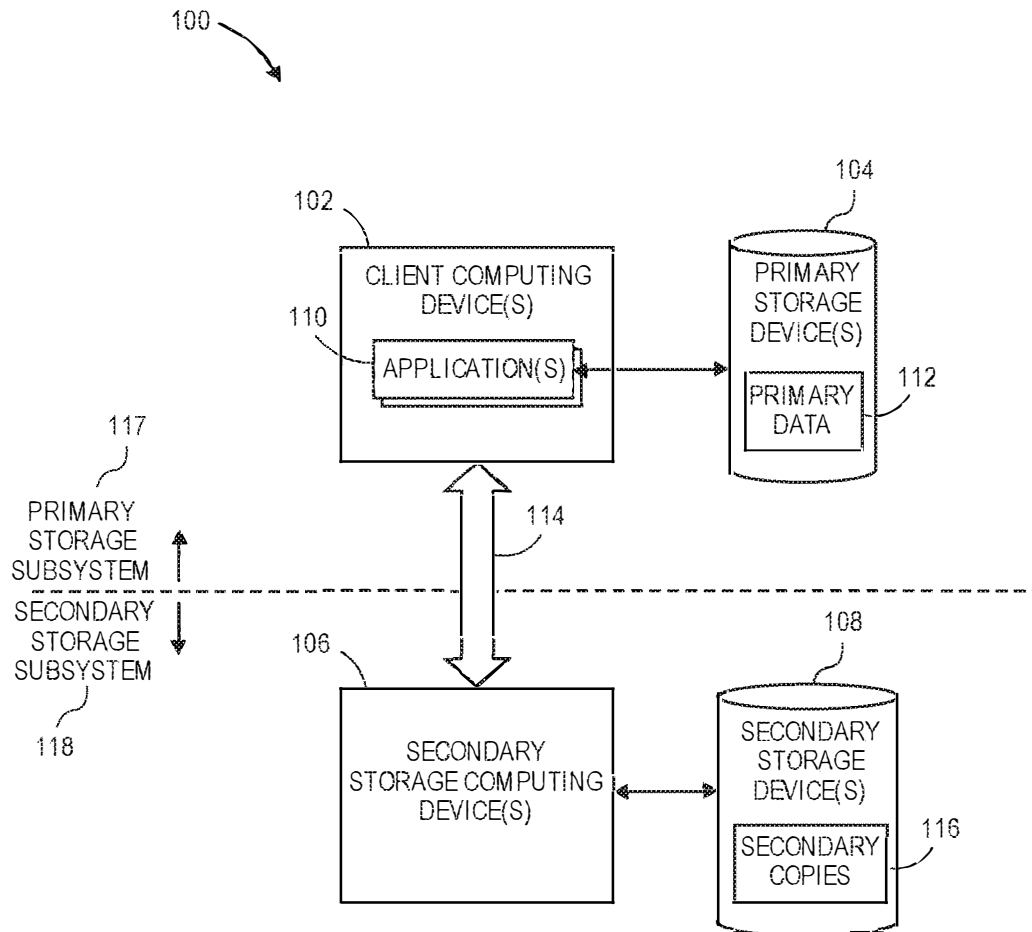
Cabrera, et al. "ADSM: A Multi-Platform, Scalable, Back-up and Archive Mass Storage System," Digest of Papers, Compcon '95, Proceedings of the 40th IEEE Computer Society International Conference, Mar. 5, 1995-Mar. 9, 1995, pp. 420-427, San Francisco, CA.

Eitel, "Backup and Storage Management in Distributed Heterogeneous Environments," IEEE, 1994, pp. 124-126.

Huff, KL, "Data Set Usage Sequence Number," IBM Technical Disclosure Bulletin, vol. 24, No. 5, Oct. 1981 New York, US, pp. 2404-2406.

Rosenblum et al., "The Design and Implementation of a Log-Structure File System," Operating Systems Review SIGOPS, vol. 25, No. 5, May 1991, New York, US, pp. 1-15.

* cited by examiner

**FIG. 1A**

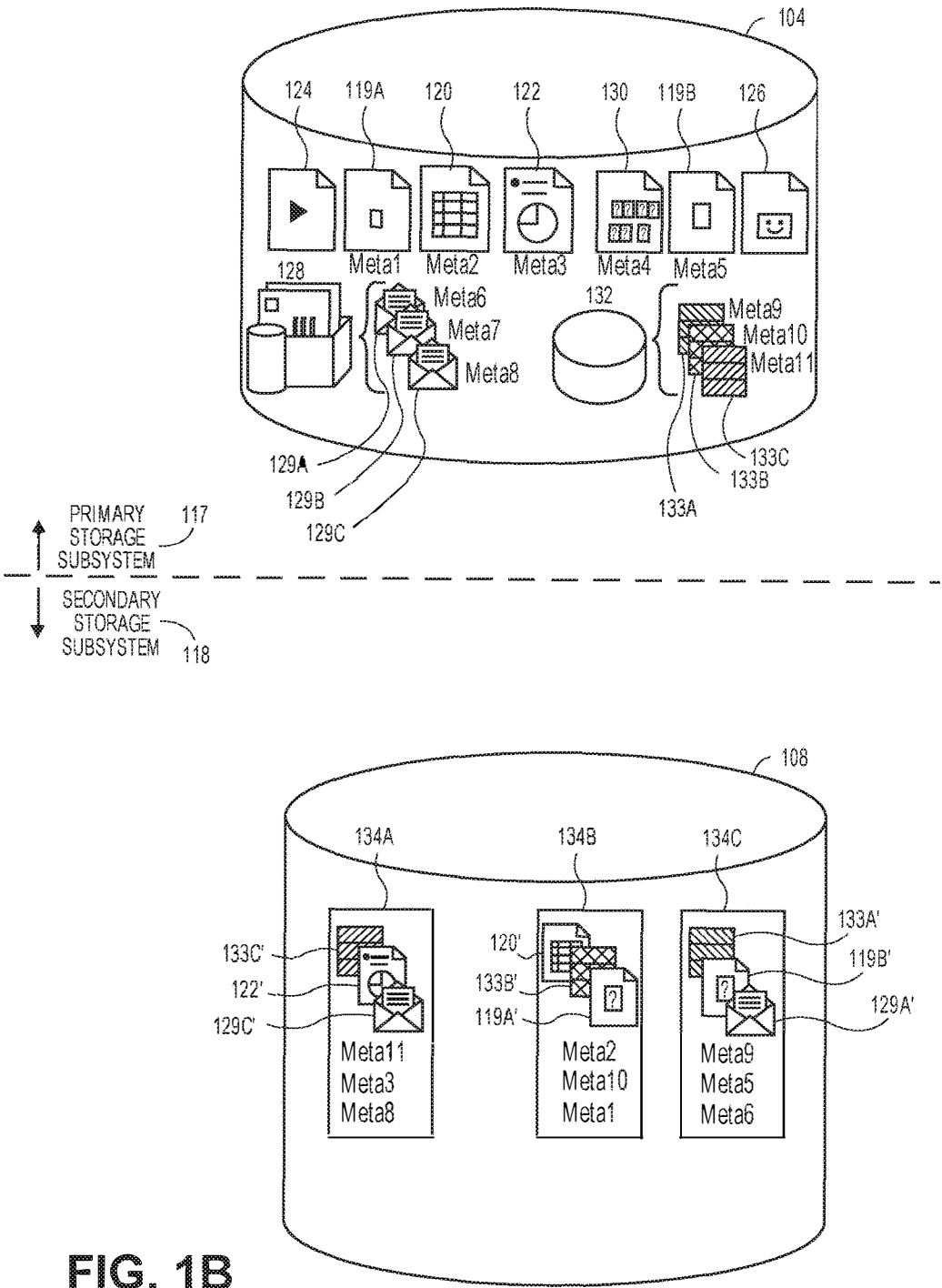


FIG. 1B

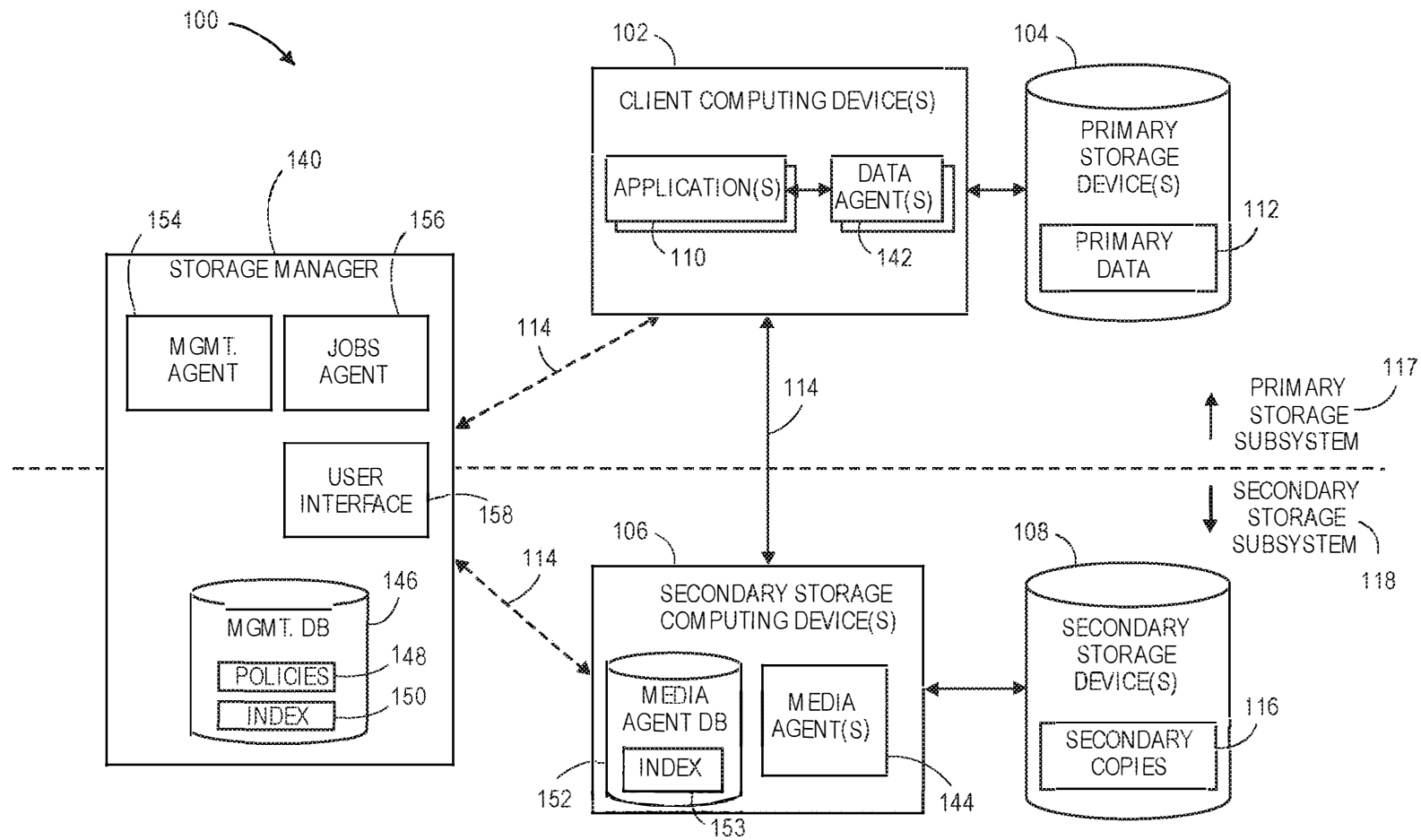


FIG. 1C

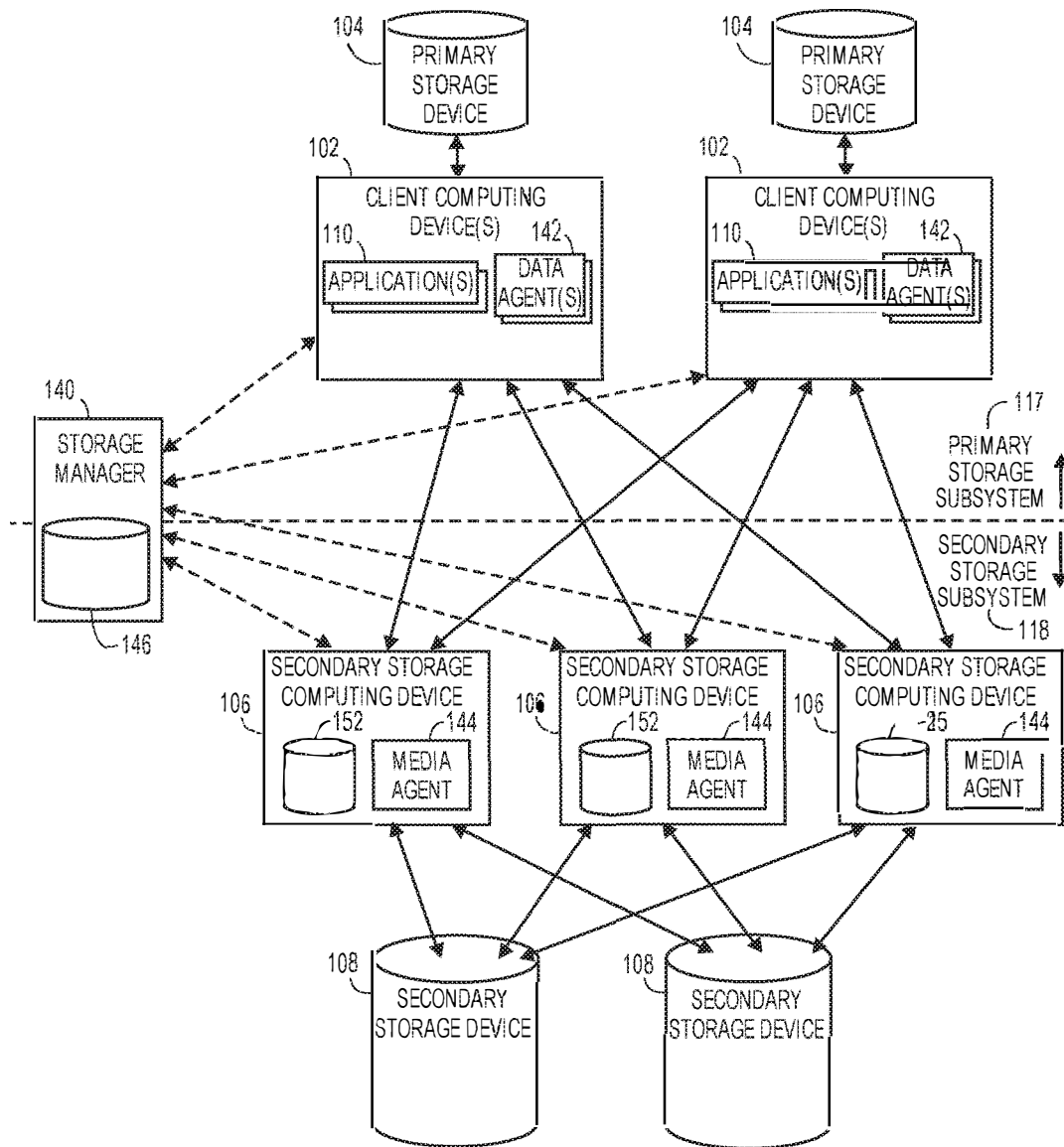


FIG. 1D

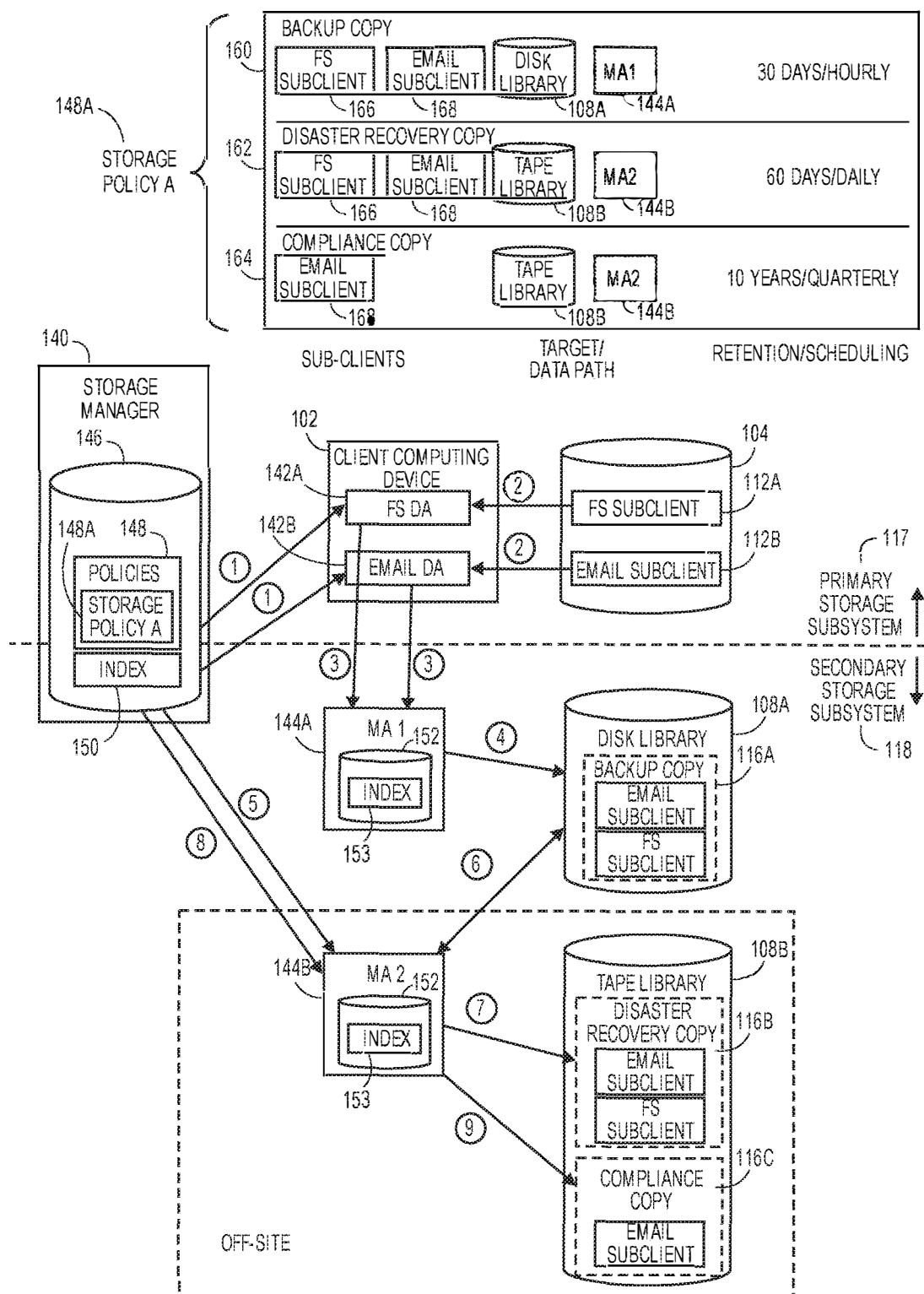


FIG. 1E

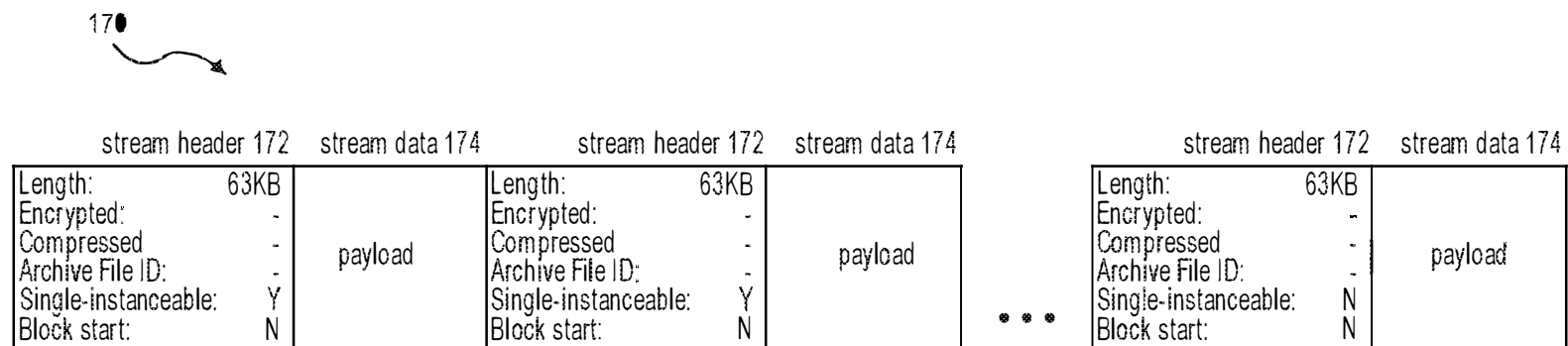


FIG. 1F

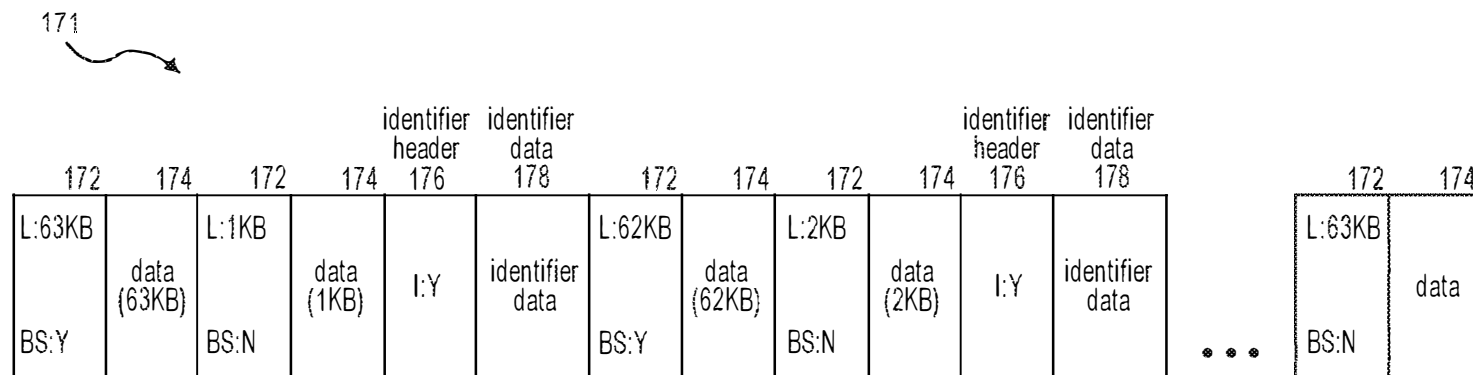


FIG. 1G

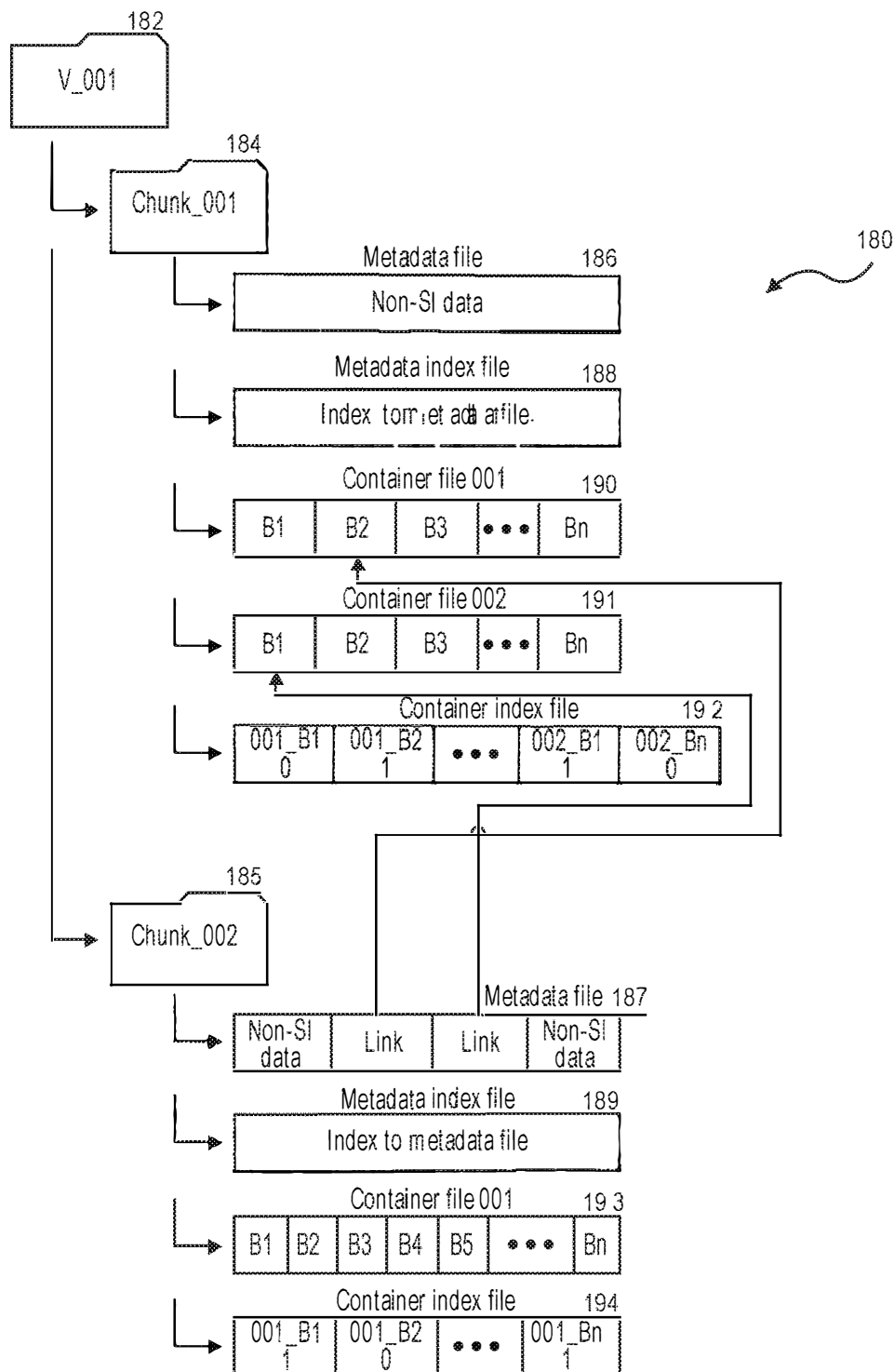


FIG. 1H

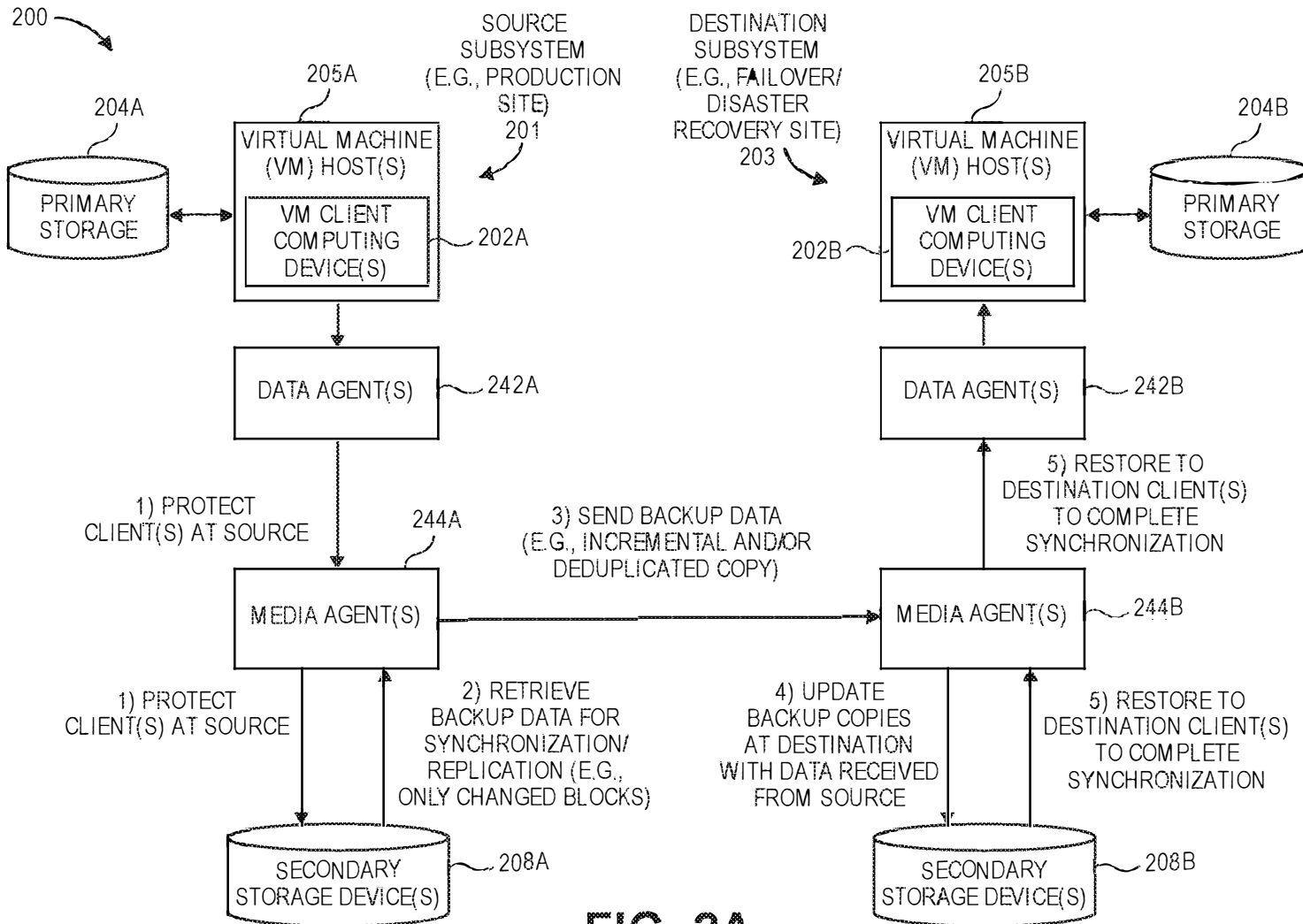


FIG. 2A

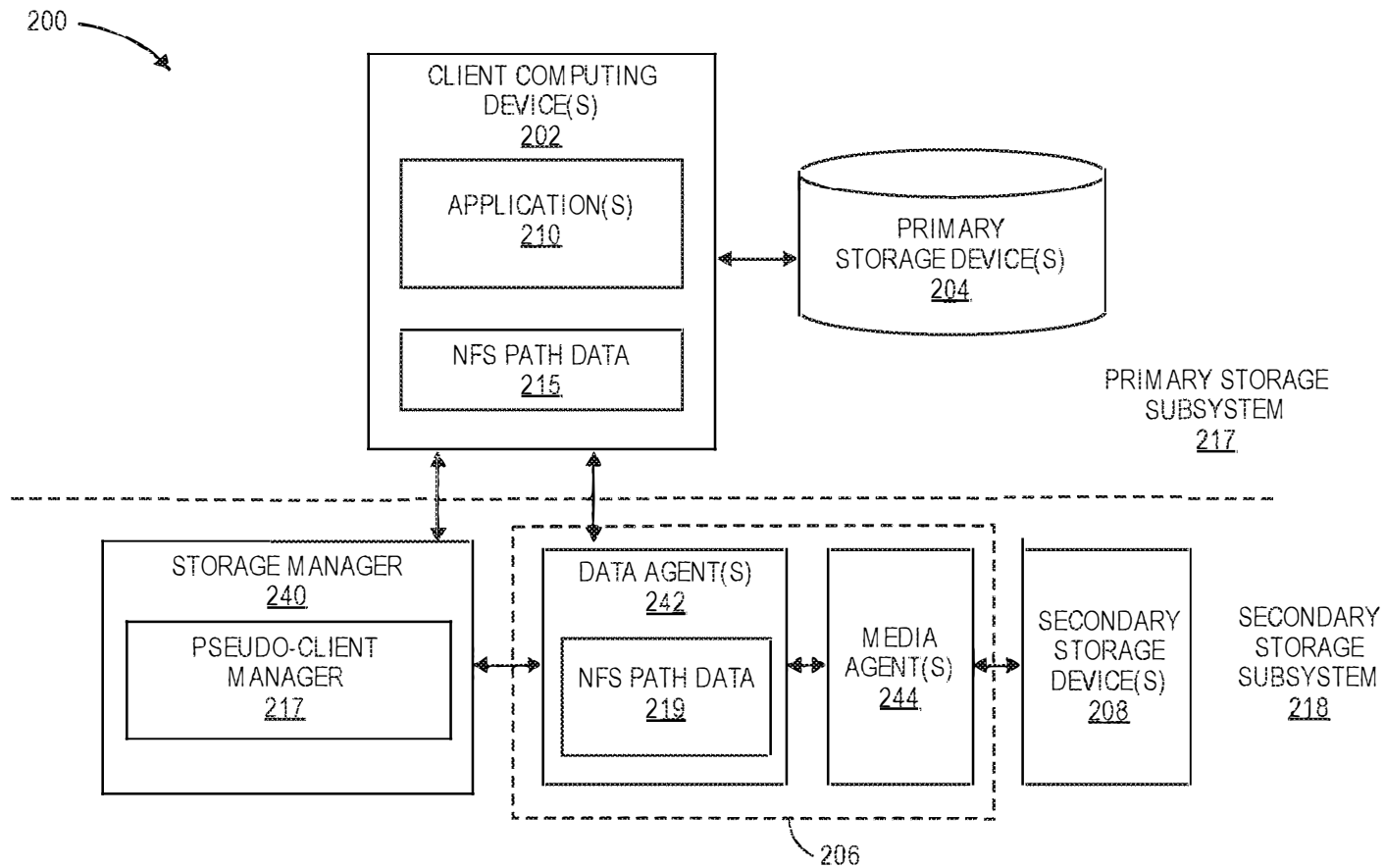


FIG. 2B

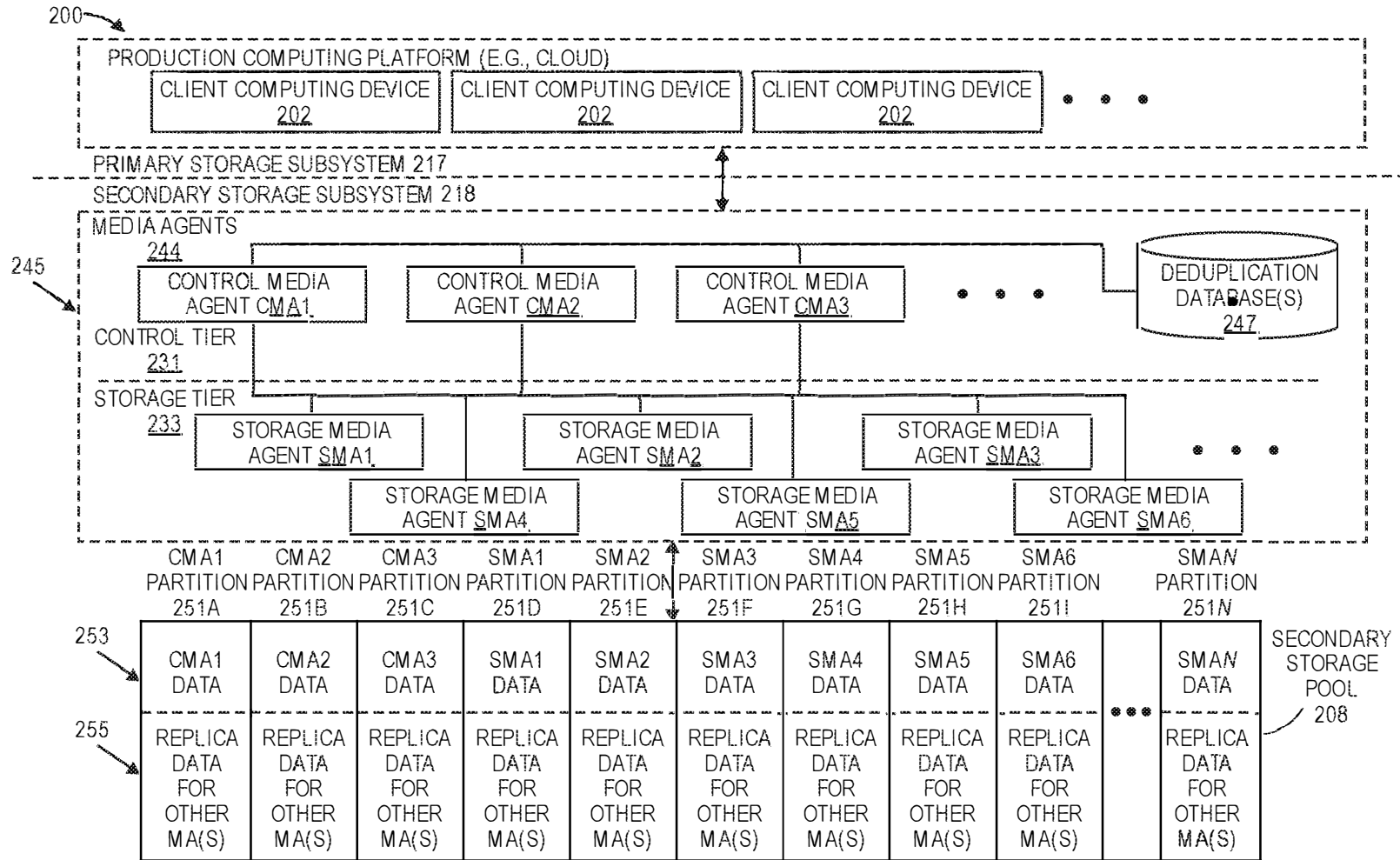


FIG. 2C

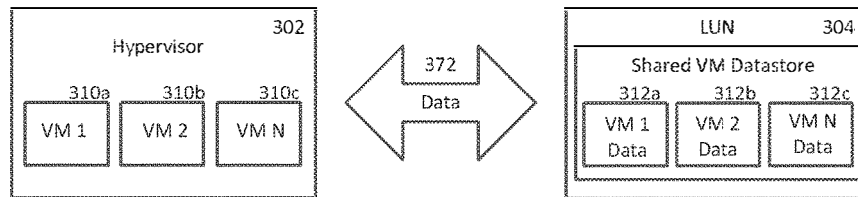
300

FIG. 3

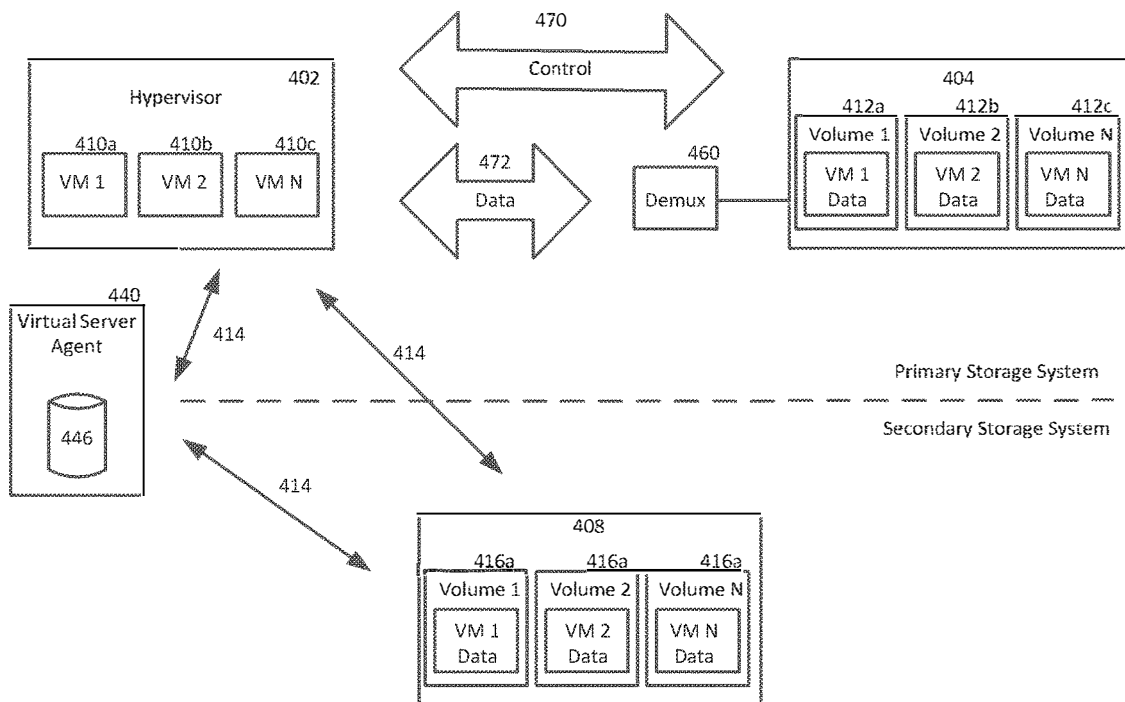
400

FIG. 4

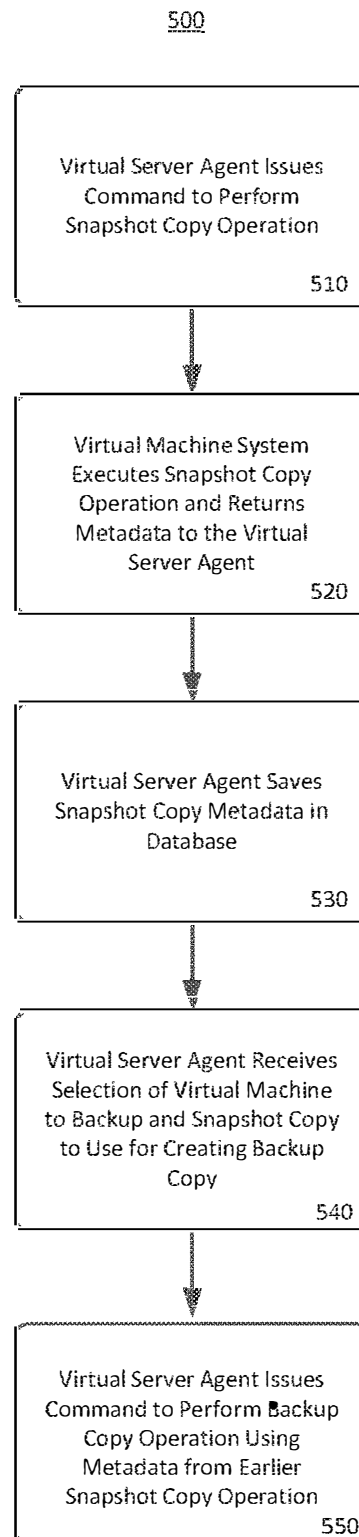


FIG. 5

600

Virtual Server Agent

Job Controller

Filters:

Snap ID	VM	Date	Time	Path
87688	VM N	Jan. 1	9:00 am	Primary Storage\Volume N\...
88134	VM 2	Jan. 1	12:00 pm	Primary Storage\Volume 2\...
88079	VM 1	Jan. 1	9:00 am	Primary Storage\Volume 1\...
88080	VM 1	Jan. 2	3:00 pm	Primary Storage\Volume 1\...

FIG. 6

US 10,210,048 B2

1

SELECTIVE SNAPSHOT AND BACKUP COPY OPERATIONS FOR INDIVIDUAL VIRTUAL MACHINES IN A SHARED STORAGE

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document and/or the patent disclosure as it appears in the United States Patent and Trademark Office patent file and/or records, but otherwise reserves all copyrights whatsoever.

BACKGROUND

Businesses recognize the commercial value of their data and seek reliable, cost-effective ways to protect the information stored on their computer networks while minimizing impact on productivity. A company might back up critical computing systems such as databases, file servers, web servers, virtual machines, and so on as part of a daily, weekly, or monthly maintenance schedule. The company may similarly protect computing systems used by its employees, such as those used by an accounting department, marketing department, engineering department, and so forth. Given the rapidly expanding volume of data under management, companies also continue to seek innovative techniques for managing data growth, for example by migrating data to lower-cost storage over time, reducing redundant data, pruning lower priority data, etc. Enterprises also increasingly view their stored data as a valuable asset and look for solutions that leverage their data. For instance, data analysis capabilities, information management, improved data presentation and access features, and the like, are in increasing demand.

SUMMARY

In some embodiments, a system comprises: a hypervisor configured to create and operate a plurality of virtual machines; one or more shared physical computer storage devices communicatively coupled to the hypervisor to store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; and a virtual server agent configured to issue a command to the hypervisor to perform a snapshot copy operation for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

In some embodiments, a method comprises: by a hypervisor, creating and operating a plurality of virtual machines; by one or more shared physical computer storage devices communicatively coupled to the hypervisor, storing the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; and by a virtual server agent, issuing a command to the hypervisor to perform a snapshot copy operation for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

2

In some embodiments, a virtual server agent comprises: a memory for storing instructions to carry out a method comprising issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; receiving metadata corresponding to the completed snapshot copy operation from the hypervisor; and storing the metadata in a database, wherein the snapshot copy operation is performed for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices; and a processor to execute the instructions.

In some embodiments, a virtual server agent method comprises: by the virtual server agent, issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; by the virtual server agent, receiving metadata corresponding to the completed snapshot copy operation from the hypervisor; and by the virtual server agent, storing the metadata in a database, wherein the snapshot copy operation is performed for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

In some embodiments, a non-transitory computer readable medium stores instructions which when executed by at least one computing device perform a virtual server agent method comprising: by the virtual server agent, issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; by the virtual server agent, receiving metadata corresponding to the completed snapshot copy operation from the hypervisor; and by the virtual server agent, storing the metadata in a database, wherein the snapshot copy operation is performed for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

In some embodiments, a virtual server agent comprises: a memory for storing instructions to carry out a method comprising commanding an external system to create a snapshot copy of any one of a plurality of virtual machines stored in the external system; receiving metadata regarding the completed snapshot copy from the external system; storing the metadata in a database; receiving a selection of any of one or more completed snapshot copies to use for creating a backup copy of any of the plurality of virtual machines; and commanding the external system to create a

US 10,210,048 B2

3

backup copy of a selected virtual machine using metadata corresponding to the selected snapshot; and a processor to execute the instructions.

In some embodiments, a method performed by a virtual server agent comprises: by the virtual server agent, commanding an external system to create a snapshot copy of any one of a plurality of virtual machines stored in the external system; by the virtual server agent, receiving metadata regarding the completed snapshot copy from the external system; by the virtual server agent, storing the metadata in a database; by the virtual server agent, receiving a selection of any of one or more completed snapshot copies to use for creating a backup copy of any of the plurality of virtual machines; and by the virtual server agent, commanding the external system to create a backup copy of a selected virtual machine using metadata corresponding to the selected snapshot.

In some embodiments, a non-transitory computer readable medium stores instructions which when executed by at least one computing device perform a virtual server agent method comprising: by the virtual server agent, commanding an external system to create a snapshot copy of any one of a plurality of virtual machines stored in the external system; by the virtual server agent, receiving metadata regarding the completed snapshot copy from the external system; by the virtual server agent, storing the metadata in a database; by the virtual server agent, receiving a selection of any of one or more completed snapshot copies to use for creating a backup copy of any of the plurality of virtual machines; and by the virtual server agent, commanding the external system to create a backup copy of a selected virtual machine using metadata corresponding to the selected snapshot.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A is a block diagram illustrating an exemplary information management system.

FIG. 1B is a detailed view of a primary storage device, a secondary storage device, and some examples of primary data and secondary copy data.

FIG. 1C is a block diagram of an exemplary information management system including a storage manager, one or more data agents, and one or more media agents.

FIG. 1D is a block diagram illustrating a scalable information management system.

FIG. 1E illustrates certain secondary copy operations according to an exemplary storage policy.

FIGS. 1F-1H are block diagrams illustrating suitable data structures that may be employed by the information management system.

FIG. 2A illustrates a system and technique for synchronizing primary data to a destination such as a failover site using secondary copy data.

FIG. 2B illustrates an information management system architecture incorporating use of a network file system (NFS) protocol for communicating between the primary and secondary storage subsystems.

FIG. 2C is a block diagram of an example of a highly scalable managed data pool architecture.

FIG. 3 illustrates an example virtual machine system.

FIG. 4 is a block diagram illustrating some salient portions of a system for selectively and individually making secondary copies of virtual machines in a shared storage, according to an illustrative embodiment of the present invention.

4

FIG. 5 depicts some salient operations of a method according to an illustrative embodiment of the present invention.

FIG. 6 depicts an illustrative graphical user interface showing snapshot metadata collected by the virtual server agent in the illustrative system of FIG. 4.

DETAILED DESCRIPTION

Many enterprises use virtual machines as a tool to efficiently utilize computing resources. Typically, many distinct and independent virtual machines are stored in a shared datastore. For example, hundreds or thousands of virtual machines can be stored to a single Logical Unit Number (LUN) in a storage system. As with any other data, enterprises have a need to create snapshots and backup copies of their virtual machine data. But in conventional virtual machine systems there may be difficulties with selectively making snapshots or backup copies of individual virtual machine disks in the shared datastore. For example, some systems only have the capability to perform a snapshot at the LUN level, which results in a snapshot of not just a selected virtual machine but rather all of the virtual machines stored to the same LUN. This is undesirable because it results in inefficient snapshot and backup copy operations as they pertain to virtual machine systems. Systems and techniques described herein advantageously overcome this problem and allow for snapshot and backup copy operations to be selectively performed for individual virtual machines in a shared storage.

Detailed descriptions and examples of systems and methods according to one or more illustrative embodiments of the present invention may be found in the section entitled Selective Snapshot and Backup Copy Operations for Individual Virtual Machines in a Shared Storage, as well as in the section entitled Example Embodiments, and also in FIGS. 4-6 herein. Furthermore, components and functionality for selective snapshot and backup copy operations for individual virtual machines in a shared storage may be configured and/or incorporated into information management systems such as those described herein in FIGS. 1A-1H and 2A-2C.

Various embodiments described herein are intimately tied to, enabled by, and would not exist except for, computer technology. For example, the systems and techniques for performing selective snapshot and backup copy operations for individual virtual machines in a shared storage described herein in reference to various embodiments cannot reasonably be performed by humans alone, without the computer technology upon which they are implemented.

Information Management System Overview

With the increasing importance of protecting and leveraging data, organizations simply cannot risk losing critical data. Moreover, runaway data growth and other modern realities make protecting and managing data increasingly difficult. There is therefore a need for efficient, powerful, and user-friendly solutions for protecting and managing data and for smart and efficient management of data storage. Depending on the size of the organization, there may be many data production sources which are under the purview of tens, hundreds, or even thousands of individuals. In the past, individuals were sometimes responsible for managing and protecting their own data, and a patchwork of hardware and software point solutions may have been used in any given organization. These solutions were often provided by different vendors and had limited or no interoperability. Certain embodiments described herein address these and other short-

US 10,210,048 B2

5

comings of prior approaches by implementing scalable, unified, organization-wide information management, including data storage management.

FIG. 1A shows one such information management system **100** (or “system **100**”), which generally includes combinations of hardware and software configured to protect and manage data and metadata that are generated and used by computing devices in system **100**. System **100** may be referred to in some embodiments as a “storage management system” or a “data storage management system.” System **100** performs information management operations, some of which may be referred to as “storage operations” or “data storage operations,” to protect and manage the data residing in and/or managed by system **100**. The organization that employs system **100** may be a corporation or other business entity, non-profit organization, educational institution, household, governmental agency, or the like.

Generally, the systems and associated components described herein may be compatible with and/or provide some or all of the functionality of the systems and corresponding components described in one or more of the following U.S. patents/publications and patent applications assigned to Commvault Systems, Inc., each of which is hereby incorporated by reference in its entirety herein:

- U.S. Pat. No. 7,035,880, entitled “Modular Backup and Retrieval System Used in Conjunction With a Storage Area Network”;
- U.S. Pat. No. 7,107,298, entitled “System And Method For Archiving Objects In An Information Store”;
- U.S. Pat. No. 7,246,207, entitled “System and Method for Dynamically Performing Storage Operations in a Computer Network”;
- U.S. Pat. No. 7,315,923, entitled “System And Method For Combining Data Streams In Pipelined Storage Operations In A Storage Network”;
- U.S. Pat. No. 7,343,453, entitled “Hierarchical Systems and Methods for Providing a Unified View of Storage Information”;
- U.S. Pat. No. 7,395,282, entitled “Hierarchical Backup and Retrieval System”;
- U.S. Pat. No. 7,529,782, entitled “System and Methods for Performing a Snapshot and for Restoring Data”;
- U.S. Pat. No. 7,617,262, entitled “System and Methods for Monitoring Application Data in a Data Replication System”;
- U.S. Pat. No. 7,734,669, entitled “Managing Copies Of Data”;
- U.S. Pat. No. 7,747,579, entitled “Metabase for Facilitating Data Classification”;
- U.S. Pat. No. 8,156,086, entitled “Systems And Methods For Stored Data Verification”;
- U.S. Pat. No. 8,170,995, entitled “Method and System for Offline Indexing of Content and Classifying Stored Data”;
- U.S. Pat. No. 8,230,195, entitled “System And Method For Performing Auxiliary Storage Operations”;
- U.S. Pat. No. 8,285,681, entitled “Data Object Store and Server for a Cloud Storage Environment, Including Data Deduplication and Data Management Across Multiple Cloud Storage Sites”;
- U.S. Pat. No. 8,307,177, entitled “Systems And Methods For Management Of Virtualization Data”;
- U.S. Pat. No. 8,364,652, entitled “Content-Aligned, Block-Based Deduplication”;
- U.S. Pat. No. 8,578,120, entitled “Block-Level Single Instancing”;

6

- U.S. Pat. No. 8,954,446, entitled “Client-Side Repository in a Networked Deduplicated Storage System”;
- U.S. Pat. No. 9,020,900, entitled “Distributed Deduplicated Storage System”;
- U.S. Pat. No. 9,098,495, entitled “Application-Aware and Remote Single Instance Data Management”;
- U.S. Pat. No. 9,239,687, entitled “Systems and Methods for Retaining and Using Data Block Signatures in Data Protection Operations”;
- U.S. Pat. Pub. No. 2006/0224846, entitled “System and Method to Support Single Instance Storage Operations”;
- U.S. Pat. Pub. No. 2014/0201170, entitled “High Availability Distributed Deduplicated Storage System”;
- U.S. patent application Ser. No. 14/721,971, entitled “Replication Using Deduplicated Secondary Copy Data”;
- U.S. Patent Application No. 62/265,339 entitled “Live Synchronization and Management of Virtual Machines across Computing and Virtualization Platforms and Using Live Synchronization to Support Disaster Recovery”;
- U.S. Patent Application No. 62/273,286 entitled “Redundant and Robust Distributed Deduplication Data Storage System”;
- U.S. Patent Application No. 62/294,920, entitled “Data Protection Operations Based on Network Path Information”;
- U.S. Patent Application No. 62/297,057, entitled “Data Restoration Operations Based on Network Path Information”; and
- U.S. Patent Application No. 62/387,384, entitled “Application-Level Live Synchronization Across Computing Platforms Including Synchronizing Co-Resident Applications To Disparate Standby Destinations And Selectively Synchronizing Some Applications And Not Others”.

System **100** includes computing devices and computing technologies. For instance, system **100** can include one or more client computing devices **102** and secondary storage computing devices **106**, as well as storage manager **140** or a host computing device for it. Computing devices can include, without limitation, one or more: workstations, personal computers, desktop computers, or other types of generally fixed computing systems such as mainframe computers, servers, and minicomputers. Other computing devices can include mobile or portable computing devices, such as one or more laptops, tablet computers, personal data assistants, mobile phones (such as smartphones), and other mobile or portable computing devices such as embedded computers, set top boxes, vehicle-mounted devices, wearable computers, etc. Servers can include mail servers, file servers, database servers, virtual machine servers, and web servers. Any given computing device comprises one or more processors (e.g., CPU and/or single-core or multi-core processors), as well as corresponding non-transitory computer memory (e.g., random-access memory (RAM)) for storing computer programs which are to be executed by the one or more processors. Other computer memory for mass storage of data may be packaged/configured with the computing device (e.g., an internal hard disk) and/or may be external and accessible by the computing device (e.g., network-attached storage, a storage array, etc.). In some cases, a computing device includes cloud computing resources, which may be implemented as virtual machines. For instance, one or more virtual machines may be provided to the organization by a third-party cloud service vendor.

US 10,210,048 B2

7

In some embodiments, computing devices can include one or more virtual machine(s) running on a physical host computing device (or “host machine”) operated by the organization. As one example, the organization may use one virtual machine as a database server and another virtual machine as a mail server, both virtual machines operating on the same host machine. A Virtual machine (“VM”) is a software implementation of a computer that does not physically exist and is instead instantiated in an operating system of a physical computer (or host machine) to enable applications to execute within the VM’s environment, i.e., a VM emulates a physical computer. A VM includes an operating system and associated virtual resources, such as computer memory and processor(s). A hypervisor operates between the VM and the hardware of the physical host machine and is generally responsible for creating and running the VMs. Hypervisors are also known in the art as virtual machine monitors or a virtual machine managers or “VMMs”, and may be implemented in software, firmware, and/or specialized hardware installed on the host machine. Examples of hypervisors include ESX Server, by VMware, Inc. of Palo Alto, Calif.; Microsoft Virtual Server and Microsoft Windows Server Hyper-V, both by Microsoft Corporation of Redmond, Wash.; Sun xVM by Oracle America Inc. of Santa Clara, Calif.; and Xen by Citrix Systems, Santa Clara, Calif. The hypervisor provides resources to each virtual operating system such as a virtual processor, virtual memory, a virtual network device, and a virtual disk. Each virtual machine has one or more associated virtual disks. The hypervisor typically stores the data of virtual disks in files on the file system of the physical host machine, called virtual machine disk files (“VMDK” in VMware lingo) or virtual hard disk image files (in Microsoft lingo). For example, VMware’s ESX Server provides the Virtual Machine File System (VMFS) for the storage of virtual machine disk files. A virtual machine reads data from and writes data to its virtual disk much the way that a physical machine reads data from and writes data to a physical disk. Examples of techniques for implementing information management in a cloud computing environment are described in U.S. Pat. No. 8,285,681. Examples of techniques for implementing information management in a virtualized computing environment are described in U.S. Pat. No. 8,307,177.

Information management system **100** can also include electronic data storage devices, generally used for mass storage of data, including, e.g., primary storage devices **104** and secondary storage devices **108**. Storage devices can generally be of any suitable type including, without limitation, disk drives, storage arrays (e.g., storage-area network (SAN) and/or network-attached storage (NAS) technology), semiconductor memory (e.g., solid state storage devices), network attached storage (NAS) devices, tape libraries, or other magnetic, non-tape storage devices, optical media storage devices, DNA/RNA-based memory technology, combinations of the same, etc. In some embodiments, storage devices form part of a distributed file system. In some cases, storage devices are provided in a cloud storage environment (e.g., a private cloud or one operated by a third-party vendor), whether for primary data or secondary copies or both.

Depending on context, the term “information management system” can refer to generally all of the illustrated hardware and software components in FIG. 1C, or the term may refer to only a subset of the illustrated components. For instance, in some cases, system **100** generally refers to a combination of specialized components used to protect, move, manage, manipulate, analyze, and/or process data and

8

metadata generated by client computing devices **102**. However, system **100** in some cases does not include the underlying components that generate and/or store primary data **112**, such as the client computing devices **102** themselves, and the primary storage devices **104**. Likewise secondary storage devices **108** (e.g., a third-party provided cloud storage environment) may not be part of system **100**. As an example, “information management system” or “storage management system” may sometimes refer to one or more of the following components, which will be described in further detail below: storage manager, data agent, and media agent.

One or more client computing devices **102** may be part of system **100**, each client computing device **102** having an operating system and at least one application **110** and one or more accompanying data agents executing thereon; and associated with one or more primary storage devices **104** storing primary data **112**. Client computing device(s) **102** and primary storage devices **104** may generally be referred to in some cases as primary storage subsystem **117**. Client Computing Devices, Clients, and Subclients

Typically, a variety of sources in an organization produce data to be protected and managed. As just one illustrative example, in a corporate environment such data sources can be employee workstations and company servers such as a mail server, a web server, a database server, a transaction server, or the like. In system **100**, data generation sources include one or more client computing devices **102**. A computing device that has a data agent **142** installed and operating on it is generally referred to as a “client computing device” **102**, and may include any type of computing device, without limitation. A client computing device **102** may be associated with one or more users and/or user accounts.

A “client” is a logical component of information management system **100**, which may represent a logical grouping of one or more data agents installed on a client computing device **102**. Storage manager **140** recognizes a client as a component of system **100**, and in some embodiments, may automatically create a client component the first time a data agent **142** is installed on a client computing device **102**. Because data generated by executable component(s) **110** is tracked by the associated data agent **142** so that it may be properly protected in system **100**, a client may be said to generate data and to store the generated data to primary storage, such as primary storage device **104**. However, the terms “client” and “client computing device” as used herein do not imply that a client computing device **102** is necessarily configured in the client/server sense relative to another computing device such as a mail server, or that a client computing device **102** cannot be a server in its own right. As just a few examples, a client computing device **102** can be and/or include mail servers, file servers, database servers, virtual machine servers, and/or web servers.

Each client computing device **102** may have application(s) **110** executing thereon which generate and manipulate the data that is to be protected from loss and managed in system **100**. Applications **110** generally facilitate the operations of an organization, and can include, without limitation, mail server applications (e.g., Microsoft Exchange Server), file system applications, mail client applications (e.g., Microsoft Exchange Client), database applications or database management systems (e.g., SQL, Oracle, SAP, Lotus Notes Database), word processing applications (e.g., Microsoft Word), spreadsheet applications, financial applications, presentation applications, graphics and/or video applications, browser applications, mobile applications, entertainment applications, and so on. Each applica-

US 10,210,048 B2

9

tion 110 may be accompanied by an application-specific data agent 142, though not all data agents 142 are application-specific or associated with only application. A file system, e.g., Microsoft Windows Explorer, may be considered an application 110 and may be accompanied by its own data agent 142. Client computing devices 102 can have at least one operating system (e.g., Microsoft Windows, Mac OS X, iOS, IBM z/OS, Linux, other Unix-based operating systems, etc.) installed thereon, which may support or host one or more file systems and other applications 110. In some embodiments, a virtual machine that executes on a host client computing device 102 may be considered an application 110 and may be accompanied by a specific data agent 142 (e.g., virtual server data agent).

Client computing devices 102 and other components in system 100 can be connected to one another via one or more electronic communication pathways 114. For example, a first communication pathway 114 may communicatively couple client computing device 102 and secondary storage computing device 106; a second communication pathway 114 may communicatively couple storage manager 140 and client computing device 102; and a third communication pathway 114 may communicatively couple storage manager 140 and secondary storage computing device 106, etc. (see, e.g., FIG. 1A and FIG. 1C). A communication pathway 114 can include one or more networks or other connection types including one or more of the following, without limitation: the Internet, a wide area network (WAN), a local area network (LAN), a Storage Area Network (SAN), a Fibre Channel (FC) connection, a Small Computer System Interface (SCSI) connection, a virtual private network (VPN), a token ring or TCP/IP based network, an intranet network, a point-to-point link, a cellular network, a wireless data transmission system, a two-way cable system, an interactive kiosk network, a satellite network, a broadband network, a baseband network, a neural network, a mesh network, an ad hoc network, other appropriate computer or telecommunications networks, combinations of the same or the like. Communication pathways 114 in some cases may also include application programming interfaces (APIs) including, e.g., cloud service provider APIs, virtual machine management APIs, and hosted service provider APIs. The underlying infrastructure of communication pathways 114 may be wired and/or wireless, analog and/or digital, or any combination thereof; and the facilities used may be private, public, third-party provided, or any combination thereof, without limitation.

A “subclient” is a logical grouping of all or part of a client’s primary data 112. In general, a subclient may be defined according to how the subclient data is to be protected as a unit in system 100. For example, a subclient may be associated with a certain storage policy. A given client may thus comprise several subclients, each subclient associated with a different storage policy. For example, some files may form a first subclient that requires compression and deduplication and is associated with a first storage policy. Other files of the client may form a second subclient that requires a different retention schedule as well as encryption, and may be associated with a different, second storage policy. As a result, though the primary data may be generated by the same application 110 and may belong to one given client, portions of the data may be assigned to different subclients for distinct treatment by system 100. More detail on subclients is given in regard to storage policies below.

Primary Data and Exemplary Primary Storage Devices

Primary data 112 is generally production data or “live” data generated by the operating system and/or applications

10

110 executing on client computing device 102. Primary data 112 is generally stored on primary storage device(s) 104 and is organized via a file system operating on the client computing device 102. Thus, client computing device(s) 102 and corresponding applications 110 may create, access, modify, write, delete, and otherwise use primary data 112. Primary data 112 is generally in the native format of the source application 110. Primary data 112 is an initial or first stored body of data generated by the source application 110. Primary data 112 in some cases is created substantially directly from data generated by the corresponding source application 110. It can be useful in performing certain tasks to organize primary data 112 into units of different granularities. In general, primary data 112 can include files, directories, file system volumes, data blocks, extents, or any other hierarchies or organizations of data objects. As used herein, a “data object” can refer to (i) any file that is currently addressable by a file system or that was previously addressable by the file system (e.g., an archive file), and/or to (ii) a subset of such a file (e.g., a data block, an extent, etc.). Primary data 112 may include structured data (e.g., database files), unstructured data (e.g., documents), and/or semi-structured data. See, e.g., FIG. 1B.

It can also be useful in performing certain functions of system 100 to access and modify metadata within primary data 112. Metadata generally includes information about data objects and/or characteristics associated with the data objects. For simplicity herein, it is to be understood that, unless expressly stated otherwise, any reference to primary data 112 generally also includes its associated metadata, but references to metadata generally do not include the primary data. Metadata can include, without limitation, one or more of the following: the data owner (e.g., the client or user that generates the data), the last modified time (e.g., the time of the most recent modification of the data object), a data object name (e.g., a file name), a data object size (e.g., a number of bytes of data), information about the content (e.g., an indication as to the existence of a particular search term), user-supplied tags, to/from information for email (e.g., an email sender, recipient, etc.), creation date, file type (e.g., format or application type), last accessed time, application type (e.g., type of application that generated the data object), location/network (e.g., a current, past or future location of the data object and network pathways to/from the data object), geographic location (e.g., GPS coordinates), frequency of change (e.g., a period in which the data object is modified), business unit (e.g., a group or department that generates, manages or is otherwise associated with the data object), aging information (e.g., a schedule, such as a time period, in which the data object is migrated to secondary or long term storage), boot sectors, partition layouts, file location within a file folder directory structure, user permissions, owners, groups, access control lists (ACLs), system metadata (e.g., registry information), combinations of the same or other similar information related to the data object. In addition to metadata generated by or related to file systems and operating systems, some applications 110 and/or other components of system 100 maintain indices of metadata for data objects, e.g., metadata associated with individual email messages. The use of metadata to perform classification and other functions is described in greater detail below.

Primary storage devices 104 storing primary data 112 may be relatively fast and/or expensive technology (e.g., flash storage, a disk drive, a hard-disk storage array, solid state memory, etc.), typically to support high-performance live production environments. Primary data 112 may be highly changeable and/or may be intended for relatively short term

US 10,210,048 B2

11

retention (e.g., hours, days, or weeks). According to some embodiments, client computing device 102 can access primary data 112 stored in primary storage device 104 by making conventional file system calls via the operating system. Each client computing device 102 is generally associated with and/or in communication with one or more primary storage devices 104 storing corresponding primary data 112. A client computing device 102 is said to be associated with or in communication with a particular primary storage device 104 if it is capable of one or more of: routing and/or storing data (e.g., primary data 112) to the primary storage device 104, coordinating the routing and/or storing of data to the primary storage device 104, retrieving data from the primary storage device 104, coordinating the retrieval of data from the primary storage device 104, and modifying and/or deleting data in the primary storage device 104. Thus, a client computing device 102 may be said to access data stored in an associated storage device 104.

Primary storage device 104 may be dedicated or shared. In some cases, each primary storage device 104 is dedicated to an associated client computing device 102, e.g., a local disk drive. In other cases, one or more primary storage devices 104 can be shared by multiple client computing devices 102, e.g., via a local network, in a cloud storage implementation, etc. As one example, primary storage device 104 can be a storage array shared by a group of client computing devices 102, such as EMC Clariion, EMC Symmetrix, EMC Celerra, Dell EqualLogic, IBM XIV, NetApp FAS, HP EVA, and HP 3PAR.

System 100 may also include hosted services (not shown), which may be hosted in some cases by an entity other than the organization that employs the other components of system 100. For instance, the hosted services may be provided by online service providers. Such service providers can provide social networking services, hosted email services, or hosted productivity applications or other hosted applications such as software-as-a-service (SaaS), platform-as-a-service (PaaS), application service providers (ASPs), cloud services, or other mechanisms for delivering functionality via a network. As it services users, each hosted service may generate additional data and metadata, which may be managed by system 100, e.g., as primary data 112. In some cases, the hosted services may be accessed using one of the applications 110. As an example, a hosted mail service may be accessed via browser running on a client computing device 102.

Secondary Copies and Exemplary Secondary Storage Devices

Primary data 112 stored on primary storage devices 104 may be compromised in some cases, such as when an employee deliberately or accidentally deletes or overwrites primary data 112. Or primary storage devices 104 can be damaged, lost, or otherwise corrupted. For recovery and/or regulatory compliance purposes, it is therefore useful to generate and maintain copies of primary data 112. Accordingly, system 100 includes one or more secondary storage computing devices 106 and one or more secondary storage devices 108 configured to create and store one or more secondary copies 116 of primary data 112 including its associated metadata. The secondary storage computing devices 106 and the secondary storage devices 108 may be referred to as secondary storage subsystem 118.

Secondary copies 116 can help in search and analysis efforts and meet other information management goals as well, such as: restoring data and/or metadata if an original version is lost (e.g., by deletion, corruption, or disaster); allowing point-in-time recovery; complying with regulatory

12

data retention and electronic discovery (e-discovery) requirements; reducing utilized storage capacity in the production system and/or in secondary storage; facilitating organization and search of data; improving user access to data files across multiple computing devices and/or hosted services; and implementing data retention and pruning policies.

A secondary copy 116 can comprise a separate stored copy of data that is derived from one or more earlier-created stored copies (e.g., derived from primary data 112 or from another secondary copy 116). Secondary copies 116 can include point-in-time data, and may be intended for relatively long-term retention before some or all of the data is moved to other storage or discarded. In some cases, a secondary copy 116 may be in a different storage device than other previously stored copies; and/or may be remote from other previously stored copies. Secondary copies 116 can be stored in the same storage device as primary data 112. For example, a disk array capable of performing hardware snapshots stores primary data 112 and creates and stores hardware snapshots of the primary data 112 as secondary copies 116. Secondary copies 116 may be stored in relatively slow and/or lower cost storage (e.g., magnetic tape). A secondary copy 116 may be stored in a backup or archive format, or in some other format different from the native source application format or other format of primary data 112.

Secondary storage computing devices 106 may index secondary copies 116 (e.g., using a media agent 144), enabling users to browse and restore at a later time and further enabling the lifecycle management of the indexed data. After creation of a secondary copy 116 that represents certain primary data 112, a pointer or other location indicia (e.g., a stub) may be placed in primary data 112, or be otherwise associated with primary data 112, to indicate the current location of a particular secondary copy 116. Since an instance of a data object or metadata in primary data 112 may change over time as it is modified by application 110 (or hosted service or the operating system), system 100 may create and manage multiple secondary copies 116 of a particular data object or metadata, each copy representing the state of the data object in primary data 112 at a particular point in time. Moreover, since an instance of a data object in primary data 112 may eventually be deleted from primary storage device 104 and the file system, system 100 may continue to manage point-in-time representations of that data object, even though the instance in primary data 112 no longer exists. For virtual machines, the operating system and other applications 110 of client computing device(s) 102 may execute within or under the management of virtualization software (e.g., a VMM), and the primary storage device(s) 104 may comprise a virtual disk created on a physical storage device. System 100 may create secondary copies 116 of the files or other data objects in a virtual disk file and/or secondary copies 116 of the entire virtual disk file itself (e.g., of an entire .vmdk file).

Secondary copies 116 are distinguishable from corresponding primary data 112. First, secondary copies 116 can be stored in a different format from primary data 112 (e.g., backup, archive, or other non-native format). For this or other reasons, secondary copies 116 may not be directly usable by applications 110 or client computing device 102 (e.g., via standard system calls or otherwise) without modification, processing, or other intervention by system 100 which may be referred to as “restore” operations. Secondary copies 116 may have been processed by data agent 142 and/or media agent 144 in the course of being created (e.g.,

US 10,210,048 B2

13

compression, deduplication, encryption, integrity markers, indexing, formatting, application-aware metadata, etc.), and thus secondary copy 116 may represent source primary data 112 without necessarily being exactly identical to the source.

Second, secondary copies 116 may be stored on a secondary storage device 108 that is inaccessible to application 110 running on client computing device 102 and/or hosted service. Some secondary copies 116 may be “offline copies,” in that they are not readily available (e.g., not mounted to tape or disk). Offline copies can include copies of data that system 100 can access without human intervention (e.g., tapes within an automated tape library, but not yet mounted in a drive), and copies that the system 100 can access only with some human intervention (e.g., tapes located at an offsite storage site).

Using Intermediate Devices for Creating Secondary Copies—Secondary Storage Computing Devices

Creating secondary copies can be challenging when hundreds or thousands of client computing devices 102 continually generate large volumes of primary data 112 to be protected. Also, there can be significant overhead involved in the creation of secondary copies 116. Moreover, specialized programmed intelligence and/or hardware capability is generally needed for accessing and interacting with secondary storage devices 108. Client computing devices 102 may interact directly with a secondary storage device 108 to create secondary copies 116, but in view of the factors described above, this approach can negatively impact the ability of client computing device 102 to serve/service application 110 and produce primary data 112. Further, any given client computing device 102 may not be optimized for interaction with certain secondary storage devices 108.

Thus, system 100 may include one or more software and/or hardware components which generally act as intermediaries between client computing devices 102 (that generate primary data 112) and secondary storage devices 108 (that store secondary copies 116). In addition to off-loading certain responsibilities from client computing devices 102, these intermediate components provide other benefits. For instance, as discussed further below with respect to FIG. 1D, distributing some of the work involved in creating secondary copies 116 can enhance scalability and improve system performance. For instance, using specialized secondary storage computing devices 106 and media agents 144 for interfacing with secondary storage devices 108 and/or for performing certain data processing operations can greatly improve the speed with which system 100 performs information management operations and can also improve the capacity of the system to handle large numbers of such operations, while reducing the computational load on the production environment of client computing devices 102. The intermediate components can include one or more secondary storage computing devices 106 as shown in FIG. 1A and/or one or more media agents 144. Media agents are discussed further below (e.g., with respect to FIGS. 1C-1E). These special-purpose components of system 100 comprise specialized programmed intelligence and/or hardware capability for writing to, reading from, instructing, communicating with, or otherwise interacting with secondary storage devices 108.

Secondary storage computing device(s) 106 can comprise any of the computing devices described above, without limitation. In some cases, secondary storage computing device(s) 106 also include specialized hardware componentry and/or software intelligence (e.g., specialized interfaces) for interacting with certain secondary storage device(s) 108 with which they may be specially associated.

14

To create a secondary copy 116 involving the copying of data from primary storage subsystem 117 to secondary storage subsystem 118, client computing device 102 may communicate the primary data 112 to be copied (or a processed version thereof generated by a data agent 142) to the designated secondary storage computing device 106, via a communication pathway 114. Secondary storage computing device 106 in turn may further process and convey the data or a processed version thereof to secondary storage device 108. One or more secondary copies 116 may be created from existing secondary copies 116, such as in the case of an auxiliary copy operation, described further below. Exemplary Primary Data and an Exemplary Secondary Copy

FIG. 1B is a detailed view of some specific examples of primary data stored on primary storage device(s) 104 and secondary copy data stored on secondary storage device(s) 108, with other components of the system removed for the purposes of illustration. Stored on primary storage device(s) 104 are primary data 112 objects including word processing documents 119A-B, spreadsheets 120, presentation documents 122, video files 124, image files 126, email mailboxes 128 (and corresponding email messages 129A-C), HTML/XML or other types of markup language files 130, databases 132 and corresponding tables or other data structures 133A-133C. Some or all primary data 112 objects are associated with corresponding metadata (e.g., “Meta1-11”), which may include file system metadata and/or application-specific metadata. Stored on the secondary storage device(s) 108 are secondary copy 116 data objects 134A-C which may include copies of or may otherwise represent corresponding primary data 112.

Secondary copy data objects 134A-C can individually represent more than one primary data object. For example, secondary copy data object 134A represents three separate primary data objects 133C, 122, and 129C (represented as 133C', 122', and 129C', respectively, and accompanied by corresponding metadata Meta1, Meta3, and Meta8, respectively). Moreover, as indicated by the prime mark ('), secondary storage computing devices 106 or other components in secondary storage subsystem 118 may process the data received from primary storage subsystem 117 and store a secondary copy including a transformed and/or supplemented representation of a primary data object and/or metadata that is different from the original format, e.g., in a compressed, encrypted, deduplicated, or other modified format. For instance, secondary storage computing devices 106 can generate new metadata or other information based on said processing, and store the newly generated information along with the secondary copies. Secondary copy data object 134B represents primary data objects 120, 1336, and 119A as 120', 1336', and 119A', respectively, accompanied by corresponding metadata Meta2, Meta10, and Meta1, respectively. Also, secondary copy data object 134C represents primary data objects 133A, 1196, and 129A as 133A', 1196', and 129A', respectively, accompanied by corresponding metadata Meta9, Meta5, and Meta6, respectively.

Exemplary Information Management System Architecture

System 100 can incorporate a variety of different hardware and software components, which can in turn be organized with respect to one another in many different configurations, depending on the embodiment. There are critical design choices involved in specifying the functional responsibilities of the components and the role of each component in system 100. Such design choices can impact how system 100 performs and adapts to data growth and other changing circumstances. FIG. 1C shows a system 100 designed

US 10,210,048 B2

15

according to these considerations and includes: storage manager 140, one or more data agents 142 executing on client computing device(s) 102 and configured to process primary data 112, and one or more media agents 144 executing on one or more secondary storage computing devices 106 for performing tasks involving secondary storage devices 108.

Storage Manager

Storage manager 140 is a centralized storage and/or information manager that is configured to perform certain control functions and also to store certain critical information about system 100—hence storage manager 140 is said to manage system 100. As noted, the number of components in system 100 and the amount of data under management can be large. Managing the components and data is therefore a significant task, which can grow unpredictably as the number of components and data scale to meet the needs of the organization. For these and other reasons, according to certain embodiments, responsibility for controlling system 100, or at least a significant portion of that responsibility, is allocated to storage manager 140. Storage manager 140 can be adapted independently according to changing circumstances, without having to replace or re-design the remainder of the system. Moreover, a computing device for hosting and/or operating as storage manager 140 can be selected to best suit the functions and networking needs of storage manager 140. These and other advantages are described in further detail below and with respect to FIG. 1D.

Storage manager 140 may be a software module or other application hosted by a suitable computing device. In some embodiments, storage manager 140 is itself a computing device that performs the functions described herein. Storage manager 140 comprises or operates in conjunction with one or more associated data structures such as a dedicated database (e.g., management database 146), depending on the configuration. The storage manager 140 generally initiates, performs, coordinates, and/or controls storage and other information management operations performed by system 100, e.g., to protect and control primary data 112 and secondary copies 116. In general, storage manager 140 is said to manage system 100, which includes communicating with, instructing, and controlling in some circumstances components such as data agents 142 and media agents 144, etc.

As shown by the dashed arrowed lines 114 in FIG. 1C, storage manager 140 may communicate with, instruct, and/or control some or all elements of system 100, such as data agents 142 and media agents 144. In this manner, storage manager 140 manages the operation of various hardware and software components in system 100. In certain embodiments, control information originates from storage manager 140 and status as well as index reporting is transmitted to storage manager 140 by the managed components, whereas payload data and metadata are generally communicated between data agents 142 and media agents 144 (or otherwise between client computing device(s) 102 and secondary storage computing device(s) 106), e.g., at the direction of and under the management of storage manager 140. Control information can generally include parameters and instructions for carrying out information management operations, such as, without limitation, instructions to perform a task associated with an operation, timing information specifying when to initiate a task, data path information specifying what components to communicate with or access in carrying out an operation, and the like. In other embodiments, some information management operations are controlled or initiated

16

ated by other components of system 100 (e.g., by media agents 144 or data agents 142), instead of or in combination with storage manager 140.

According to certain embodiments, storage manager 140 provides one or more of the following functions:

- communicating with data agents 142 and media agents 144, including transmitting instructions, messages, and/or queries, as well as receiving status reports, index information, messages, and/or queries, and responding to same;
- initiating execution of information management operations;
- initiating restore and recovery operations;
- managing secondary storage devices 108 and inventory/capacity of the same;
- allocating secondary storage devices 108 for secondary copy operations;
- reporting, searching, and/or classification of data in system 100;
- monitoring completion of and status reporting related to information management operations and jobs;
- tracking movement of data within system 100;
- tracking age information relating to secondary copies 116, secondary storage devices 108, comparing the age information against retention guidelines, and initiating data pruning when appropriate;
- tracking logical associations between components in system 100;
- protecting metadata associated with system 100, e.g., in management database 146;
- implementing job management, schedule management, event management, alert management, reporting, job history maintenance, user security management, disaster recovery management, and/or user interfacing for system administrators and/or end users of system 100;
- sending, searching, and/or viewing of log files; and
- implementing operations management functionality.

Storage manager 140 may maintain an associated database 146 (or “storage manager database 146” or “management database 146”) of management-related data and information management policies 148. Database 146 is stored in computer memory accessible by storage manager 140. Database 146 may include a management index 150 (or “index 150”) or other data structure(s) that may store: logical associations between components of the system; user preferences and/or profiles (e.g., preferences regarding encryption, compression, or deduplication of primary data or secondary copies; preferences regarding the scheduling, type, or other aspects of secondary copy or other operations; mappings of particular information management users or user accounts to certain computing devices or other components, etc.; management tasks; media containerization; other useful data; and/or any combination thereof. For example, storage manager 140 may use index 150 to track logical associations between media agents 144 and secondary storage devices 108 and/or movement of data to/from secondary storage devices 108. For instance, index 150 may store data associating a client computing device 102 with a particular media agent 144 and/or secondary storage device 108, as specified in an information management policy 148.

Administrators and others may configure and initiate certain information management operations on an individual basis. But while this may be acceptable for some recovery operations or other infrequent tasks, it is often not workable for implementing on-going organization-wide data protection and management. Thus, system 100 may utilize information management policies 148 for specifying and execut-

US 10,210,048 B2

17

ing information management operations on an automated basis. Generally, an information management policy **148** can include a stored data structure or other information source that specifies parameters (e.g., criteria and rules) associated with storage management or other information management operations. Storage manager **140** can process an information management policy **148** and/or index **150** and, based on the results, identify an information management operation to perform, identify the appropriate components in system **100** to be involved in the operation (e.g., client computing devices **102** and corresponding data agents **142**, secondary storage computing devices **106** and corresponding media agents **144**, etc.), establish connections to those components and/or between those components, and/or instruct and control those components to carry out the operation. In this manner, system **100** can translate stored information into coordinated activity among the various computing devices in system **100**.

Management database **146** may maintain information management policies **148** and associated data, although information management policies **148** can be stored in computer memory at any appropriate location outside management database **146**. For instance, an information management policy **148** such as a storage policy may be stored as metadata in a media agent database **152** or in a secondary storage device **108** (e.g., as an archive copy) for use in restore or other information management operations, depending on the embodiment. Information management policies **148** are described further below. According to certain embodiments, management database **146** comprises a relational database (e.g., an SQL database) for tracking metadata, such as metadata associated with secondary copy operations (e.g., what client computing devices **102** and corresponding subclient data were protected and where the secondary copies are stored and which media agent **144** performed the storage operation(s)). This and other metadata may additionally be stored in other locations, such as at secondary storage computing device **106** or on the secondary storage device **108**, allowing data recovery without the use of storage manager **140** in some cases. Thus, management database **146** may comprise data needed to kick off secondary copy operations (e.g., storage policies, schedule policies, etc.), status and reporting information about completed jobs (e.g., status and error reports on yesterday's backup jobs), and additional information sufficient to enable restore and disaster recovery operations (e.g., media agent associations, location indexing, content indexing, etc.).

Storage manager **140** may include a jobs agent **156**, a user interface **158**, and a management agent **154**, all of which may be implemented as interconnected software modules or application programs. These are described further below.

Jobs agent **156** in some embodiments initiates, controls, and/or monitors the status of some or all information management operations previously performed, currently being performed, or scheduled to be performed by system **100**. A job is a logical grouping of information management operations such as daily storage operations scheduled for a certain set of subclients (e.g., generating incremental block-level backup copies **116** at a certain time every day for database files in a certain geographical location). Thus, jobs agent **156** may access information management policies **148** (e.g., in management database **146**) to determine when, where, and how to initiate/control jobs in system **100**.

Storage Manager User Interfaces

User interface **158** may include information processing and display software, such as a graphical user interface (GUI), an application program interface (API), and/or other

18

interactive interface(s) through which users and system processes can retrieve information about the status of information management operations or issue instructions to storage manager **140** and other components. Via user interface **158**, users may issue instructions to the components in system **100** regarding performance of secondary copy and recovery operations. For example, a user may modify a schedule concerning the number of pending secondary copy operations. As another example, a user may employ the GUI to view the status of pending secondary copy jobs or to monitor the status of certain components in system **100** (e.g., the amount of capacity left in a storage device). Storage manager **140** may track information that permits it to select, designate, or otherwise identify content indices, deduplication databases, or similar databases or resources or data sets within its information management cell (or another cell) to be searched in response to certain queries. Such queries may be entered by the user by interacting with user interface **158**.

Various embodiments of information management system **100** may be configured and/or designed to generate user interface data usable for rendering the various interactive user interfaces described. The user interface data may be used by system **100** and/or by another system, device, and/or software program (for example, a browser program), to render the interactive user interfaces. The interactive user interfaces may be displayed on, for example, electronic displays (including, for example, touch-enabled displays), consoles, etc., whether direct-connected to storage manager **140** or communicatively coupled remotely, e.g., via an internet connection. The present disclosure describes various embodiments of interactive and dynamic user interfaces, some of which may be generated by user interface agent **158**, and which are the result of significant technological development. The user interfaces described herein may provide improved human-computer interactions, allowing for significant cognitive and ergonomic efficiencies and advantages over previous systems, including reduced mental workloads, improved decision-making, and the like. User interface **158** may operate in a single integrated view or console (not shown). The console may support a reporting capability for generating a variety of reports, which may be tailored to a particular aspect of information management.

User interfaces are not exclusive to storage manager **140** and in some embodiments a user may access information locally from a computing device component of system **100**. For example, some information pertaining to installed data agents **142** and associated data streams may be available from client computing device **102**. Likewise, some information pertaining to media agents **144** and associated data streams may be available from secondary storage computing device **106**.

Storage Manager Management Agent

Management agent **154** can provide storage manager **140** with the ability to communicate with other components within system **100** and/or with other information management cells via network protocols and application programming interfaces (APIs) including, e.g., HTTP, HTTPS, FTP, REST, virtualization software APIs, cloud service provider APIs, and hosted service provider APIs, without limitation. Management agent **154** also allows multiple information management cells to communicate with one another. For example, system **100** in some cases may be one information management cell in a network of multiple cells adjacent to one another or otherwise logically related, e.g., in a WAN or LAN. With this arrangement, the cells may communicate with one another through respective management agents

US 10,210,048 B2

19

154. Inter-cell communications and hierarchy is described in greater detail in e.g., U.S. Pat. No. 7,343,453.

Information Management Cell

An “information management cell” (or “storage operation cell” or “cell”) may generally include a logical and/or physical grouping of a combination of hardware and software components associated with performing information management operations on electronic data, typically one storage manager 140 and at least one data agent 142 (executing on a client computing device 102) and at least one media agent 144 (executing on a secondary storage computing device 106). For instance, the components shown in FIG. 1C may together form an information management cell. Thus, in some configurations, a system 100 may be referred to as an information management cell or a storage operation cell. A given cell may be identified by the identity of its storage manager 140, which is generally responsible for managing the cell.

Multiple cells may be organized hierarchically, so that cells may inherit properties from hierarchically superior cells or be controlled by other cells in the hierarchy (automatically or otherwise). Alternatively, in some embodiments, cells may inherit or otherwise be associated with information management policies, preferences, information management operational parameters, or other properties or characteristics according to their relative position in a hierarchy of cells. Cells may also be organized hierarchically according to function, geography, architectural considerations, or other factors useful or desirable in performing information management operations. For example, a first cell may represent a geographic segment of an enterprise, such as a Chicago office, and a second cell may represent a different geographic segment, such as a New York City office. Other cells may represent departments within a particular office, e.g., human resources, finance, engineering, etc. Where delineated by function, a first cell may perform one or more first types of information management operations (e.g., one or more first types of secondary copies at a certain frequency), and a second cell may perform one or more second types of information management operations (e.g., one or more second types of secondary copies at a different frequency and under different retention rules). In general, the hierarchical information is maintained by one or more storage managers 140 that manage the respective cells (e.g., in corresponding management database(s) 146).

Data Agents

A variety of different applications 110 can operate on a given client computing device 102, including operating systems, file systems, database applications, e-mail applications, and virtual machines, just to name a few. And, as part of the process of creating and restoring secondary copies 116, the client computing device 102 may be tasked with processing and preparing the primary data 112 generated by these various applications 110. Moreover, the nature of the processing/preparation can differ across application types, e.g., due to inherent structural, state, and formatting differences among applications 110 and/or the operating system of client computing device 102. Each data agent 142 is therefore advantageously configured in some embodiments to assist in the performance of information management operations based on the type of data that is being protected at a client-specific and/or application-specific level.

Data agent 142 is a component of information system 100 and is generally directed by storage manager 140 to participate in creating or restoring secondary copies 116. Data agent 142 may be a software program (e.g., in the form of a set of executable binary files) that executes on the same

20

client computing device 102 as the associated application 110 that data agent 142 is configured to protect. Data agent 142 is generally responsible for managing, initiating, or otherwise assisting in the performance of information management operations in reference to its associated application(s) 110 and corresponding primary data 112 which is generated/accessed by the particular application(s) 110. For instance, data agent 142 may take part in copying, archiving, migrating, and/or replicating of certain primary data 112 stored in the primary storage device(s) 104. Data agent 142 may receive control information from storage manager 140, such as commands to transfer copies of data objects and/or metadata to one or more media agents 144. Data agent 142 also may compress, deduplicate, and encrypt certain primary data 112, as well as capture application-related metadata before transmitting the processed data to media agent 144. Data agent 142 also may receive instructions from storage manager 140 to restore (or assist in restoring) a secondary copy 116 from secondary storage device 108 to primary storage 104, such that the restored data may be properly accessed by application 110 in a suitable format as though it were primary data 112.

Each data agent 142 may be specialized for a particular application 110. For instance, different individual data agents 142 may be designed to handle Microsoft Exchange data, Lotus Notes data, Microsoft Windows file system data, Microsoft Active Directory Objects data, SQL Server data, SharePoint data, Oracle database data, SAP database data, virtual machines and/or associated data, and other types of data. A file system data agent, for example, may handle data files and/or other file system information. If a client computing device 102 has two or more types of data 112, a specialized data agent 142 may be used for each data type. For example, to backup, migrate, and/or restore all of the data on a Microsoft Exchange server, the client computing device 102 may use: (1) a Microsoft Exchange Mailbox data agent 142 to back up the Exchange mailboxes; (2) a Microsoft Exchange Database data agent 142 to back up the Exchange databases; (3) a Microsoft Exchange Public Folder data agent 142 to back up the Exchange Public Folders; and (4) a Microsoft Windows File System data agent 142 to back up the file system of client computing device 102. In this example, these specialized data agents 142 are treated as four separate data agents 142 even though they operate on the same client computing device 102. Other examples may include archive management data agents such as a migration archiver or a compliance archiver, Quick Recovery agents, and continuous data replication agents. Application-specific data agents 142 can provide improved performance as compared to generic agents. For instance, because application-specific data agents 142 may only handle data for a single software application, the design, operation, and performance of the data agent 142 can be streamlined. The data agent 142 may therefore execute faster and consume less persistent storage and/or operating memory than data agents designed to generically accommodate multiple different software applications 110.

Each data agent 142 may be configured to access data and/or metadata stored in the primary storage device(s) 104 associated with data agent 142 and its host client computing device 102, and process the data appropriately. For example, during a secondary copy operation, data agent 142 may arrange or assemble the data and metadata into one or more files having a certain format (e.g., a particular backup or archive format) before transferring the file(s) to a media agent 144 or other component. The file(s) may include a list of files or other metadata. In some embodiments, a data

US 10,210,048 B2

21

agent 142 may be distributed between client computing device 102 and storage manager 140 (and any other intermediate components) or may be deployed from a remote location or its functions approximated by a remote process that performs some or all of the functions of data agent 142. In addition, a data agent 142 may perform some functions provided by media agent 144. Other embodiments may employ one or more generic data agents 142 that can handle and process data from two or more different applications 110, or that can handle and process multiple data types, instead of or in addition to using specialized data agents 142. For example, one generic data agent 142 may be used to back up, migrate and restore Microsoft Exchange Mailbox data and Microsoft Exchange Database data, while another generic data agent may handle Microsoft Exchange Public Folder data and Microsoft Windows File System data.

Media Agents

As noted, off-loading certain responsibilities from client computing devices 102 to intermediate components such as secondary storage computing device(s) 106 and corresponding media agent(s) 144 can provide a number of benefits including improved performance of client computing device 102, faster and more reliable information management operations, and enhanced scalability. In one example which will be discussed further below, media agent 144 can act as a local cache of recently-copied data and/or metadata stored to secondary storage device(s) 108, thus improving restore capabilities and performance for the cached data.

Media agent 144 is a component of system 100 and is generally directed by storage manager 140 in creating and restoring secondary copies 116. Whereas storage manager 140 generally manages system 100 as a whole, media agent 144 provides a portal to certain secondary storage devices 108, such as by having specialized features for communicating with and accessing certain associated secondary storage device 108. Media agent 144 may be a software program (e.g., in the form of a set of executable binary files) that executes on a secondary storage computing device 106. Media agent 144 generally manages, coordinates, and facilitates the transmission of data between a data agent 142 (executing on client computing device 102) and secondary storage device(s) 108 associated with media agent 144. For instance, other components in the system may interact with media agent 144 to gain access to data stored on associated secondary storage device(s) 108, (e.g., to browse, read, write, modify, delete, or restore data). Moreover, media agents 144 can generate and store information relating to characteristics of the stored data and/or metadata, or can generate and store other types of information that generally provides insight into the contents of the secondary storage devices 108—generally referred to as indexing of the stored secondary copies 116. Each media agent 144 may operate on a dedicated secondary storage computing device 106, while in other embodiments a plurality of media agents 144 may operate on the same secondary storage computing device 106.

A media agent 144 may be associated with a particular secondary storage device 108 if that media agent 144 is capable of one or more of: routing and/or storing data to the particular secondary storage device 108; coordinating the routing and/or storing of data to the particular secondary storage device 108; retrieving data from the particular secondary storage device 108; coordinating the retrieval of data from the particular secondary storage device 108; and modifying and/or deleting data retrieved from the particular secondary storage device 108. Media agent 144 in certain embodiments is physically separate from the associated

22

secondary storage device 108. For instance, a media agent 144 may operate on a secondary storage computing device 106 in a distinct housing, package, and/or location from the associated secondary storage device 108. In one example, a media agent 144 operates on a first server computer and is in communication with a secondary storage device(s) 108 operating in a separate rack-mounted RAID-based system.

A media agent 144 associated with a particular secondary storage device 108 may instruct secondary storage device 108 to perform an information management task. For instance, a media agent 144 may instruct a tape library to use a robotic arm or other retrieval means to load or eject a certain storage media, and to subsequently archive, migrate, or retrieve data to or from that media, e.g., for the purpose of restoring data to a client computing device 102. As another example, a secondary storage device 108 may include an array of hard disk drives or solid state drives organized in a RAID configuration, and media agent 144 may forward a logical unit number (LUN) and other appropriate information to the array, which uses the received information to execute the desired secondary copy operation. Media agent 144 may communicate with a secondary storage device 108 via a suitable communications link, such as a SCSI or Fibre Channel link.

Each media agent 144 may maintain an associated media agent database 152. Media agent database 152 may be stored to a disk or other storage device (not shown) that is local to the secondary storage computing device 106 on which media agent 144 executes. In other cases, media agent database 152 is stored separately from the host secondary storage computing device 106. Media agent database 152 can include, among other things, a media agent index 153 (see, e.g., FIG. 1C). In some cases, media agent index 153 does not form a part of and is instead separate from media agent database 152.

Media agent index 153 (or “index 153”) may be a data structure associated with the particular media agent 144 that includes information about the stored data associated with the particular media agent and which may be generated in the course of performing a secondary copy operation or a restore. Index 153 provides a fast and efficient mechanism for locating/browsing secondary copies 116 or other data stored in secondary storage devices 108 without having to access secondary storage device 108 to retrieve the information from there. For instance, for each secondary copy 116, index 153 may include metadata such as a list of the data objects (e.g., files/subdirectories, database objects, mailbox objects, etc.), a logical path to the secondary copy 116 on the corresponding secondary storage device 108, location information (e.g., offsets) indicating where the data objects are stored in the secondary storage device 108, when the data objects were created or modified, etc. Thus, index 153 includes metadata associated with the secondary copies 116 that is readily available for use from media agent 144. In some embodiments, some or all of the information in index 153 may instead or additionally be stored along with secondary copies 116 in secondary storage device 108. In some embodiments, a secondary storage device 108 can include sufficient information to enable a “bare metal restore,” where the operating system and/or software applications of a failed client computing device 102 or another target may be automatically restored without manually reinstalling individual software packages (including operating systems).

Because index 153 may operate as a cache, it can also be referred to as an “index cache.” In such cases, information stored in index cache 153 typically comprises data that

US 10,210,048 B2

23

reflects certain particulars about relatively recent secondary copy operations. After some triggering event, such as after some time elapses or index cache 153 reaches a particular size, certain portions of index cache 153 may be copied or migrated to secondary storage device 108, e.g., on a least-recently-used basis. This information may be retrieved and uploaded back into index cache 153 or otherwise restored to media agent 144 to facilitate retrieval of data from the secondary storage device(s) 108. In some embodiments, the cached information may include format or containerization information related to archives or other files stored on storage device(s) 108.

In some alternative embodiments media agent 144 generally acts as a coordinator or facilitator of secondary copy operations between client computing devices 102 and secondary storage devices 108, but does not actually write the data to secondary storage device 108. For instance, storage manager 140 (or media agent 144) may instruct a client computing device 102 and secondary storage device 108 to communicate with one another directly. In such a case, client computing device 102 transmits data directly or via one or more intermediary components to secondary storage device 108 according to the received instructions, and vice versa. Media agent 144 may still receive, process, and/or maintain metadata related to the secondary copy operations, i.e., may continue to build and maintain index 153. In these embodiments, payload data can flow through media agent 144 for the purposes of populating index 153, but not for writing to secondary storage device 108. Media agent 144 and/or other components such as storage manager 140 may in some cases incorporate additional functionality, such as data classification, content indexing, deduplication, encryption, compression, and the like. Further details regarding these and other functions are described below.

Distributed, Scalable Architecture

As described, certain functions of system 100 can be distributed amongst various physical and/or logical components. For instance, one or more of storage manager 140, data agents 142, and media agents 144 may operate on computing devices that are physically separate from one another. This architecture can provide a number of benefits. For instance, hardware and software design choices for each distributed component can be targeted to suit its particular function. The secondary computing devices 106 on which media agents 144 operate can be tailored for interaction with associated secondary storage devices 108 and provide fast index cache operation, among other specific tasks. Similarly, client computing device(s) 102 can be selected to effectively service applications 110 in order to efficiently produce and store primary data 112.

Moreover, in some cases, one or more of the individual components of information management system 100 can be distributed to multiple separate computing devices. As one example, for large file systems where the amount of data stored in management database 146 is relatively large, database 146 may be migrated to or may otherwise reside on a specialized database server (e.g., an SQL server) separate from a server that implements the other functions of storage manager 140. This distributed configuration can provide added protection because database 146 can be protected with standard database utilities (e.g., SQL log shipping or database replication) independent from other functions of storage manager 140. Database 146 can be efficiently replicated to a remote site for use in the event of a disaster or other data loss at the primary site. Or database 146 can be replicated to another computing device within the same site, such as to a higher performance machine in the event that a storage

24

manager host computing device can no longer service the needs of a growing system 100.

The distributed architecture also provides scalability and efficient component utilization. FIG. 1D shows an embodiment of information management system 100 including a plurality of client computing devices 102 and associated data agents 142 as well as a plurality of secondary storage computing devices 106 and associated media agents 144. Additional components can be added or subtracted based on the evolving needs of system 100. For instance, depending on where bottlenecks are identified, administrators can add additional client computing devices 102, secondary storage computing devices 106, and/or secondary storage devices 108. Moreover, where multiple fungible components are available, load balancing can be implemented to dynamically address identified bottlenecks. As an example, storage manager 140 may dynamically select which media agents 144 and/or secondary storage devices 108 to use for storage operations based on a processing load analysis of media agents 144 and/or secondary storage devices 108, respectively.

Where system 100 includes multiple media agents 144 (see, e.g., FIG. 1D), a first media agent 144 may provide failover functionality for a second failed media agent 144. In addition, media agents 144 can be dynamically selected to provide load balancing. Each client computing device 102 can communicate with, among other components, any of the media agents 144, e.g., as directed by storage manager 140. And each media agent 144 may communicate with, among other components, any of secondary storage devices 108, e.g., as directed by storage manager 140. Thus, operations can be routed to secondary storage devices 108 in a dynamic and highly flexible manner, to provide load balancing, failover, etc. Further examples of scalable systems capable of dynamic storage operations, load balancing, and failover are provided in U.S. Pat. No. 7,246,207.

While distributing functionality amongst multiple computing devices can have certain advantages, in other contexts it can be beneficial to consolidate functionality on the same computing device. In alternative configurations, certain components may reside and execute on the same computing device. As such, in other embodiments, one or more of the components shown in FIG. 1C may be implemented on the same computing device. In one configuration, a storage manager 140, one or more data agents 142, and/or one or more media agents 144 are all implemented on the same computing device. In other embodiments, one or more data agents 142 and one or more media agents 144 are implemented on the same computing device, while storage manager 140 is implemented on a separate computing device, etc. without limitation.

Exemplary Types of Information Management Operations, Including Storage Operations

In order to protect and leverage stored data, system 100 can be configured to perform a variety of information management operations, which may also be referred to in some cases as storage management operations or storage operations. These operations can generally include (i) data movement operations, (ii) processing and data manipulation operations, and (iii) analysis, reporting, and management operations.

Data Movement Operations, Including Secondary Copy Operations

Data movement operations are generally storage operations that involve the copying or migration of data between different locations in system 100. For example, data movement operations can include operations in which stored data

US 10,210,048 B2

25

is copied, migrated, or otherwise transferred from one or more first storage devices to one or more second storage devices, such as from primary storage device(s) 104 to secondary storage device(s) 108, from secondary storage device(s) 108 to different secondary storage device(s) 108, from secondary storage devices 108 to primary storage devices 104, or from primary storage device(s) 104 to different primary storage device(s) 104, or in some cases within the same primary storage device 104 such as within a storage array.

Data movement operations can include by way of example, backup operations, archive operations, information lifecycle management operations such as hierarchical storage management operations, replication operations (e.g., continuous data replication), snapshot operations, deduplication or single-instancing operations, auxiliary copy operations, disaster-recovery copy operations, and the like. As will be discussed, some of these operations do not necessarily create distinct copies. Nonetheless, some or all of these operations are generally referred to as “secondary copy operations” for simplicity, because they involve secondary copies. Data movement also comprises restoring secondary copies.

Backup Operations

A backup operation creates a copy of a version of primary data 112 at a particular point in time (e.g., one or more files or other data units). Each subsequent backup copy 116 (which is a form of secondary copy 116) may be maintained independently of the first. A backup generally involves maintaining a version of the copied primary data 112 as well as backup copies 116. Further, a backup copy in some embodiments is generally stored in a form that is different from the native format, e.g., a backup format. This contrasts to the version in primary data 112 which may instead be stored in a format native to the source application(s) 110. In various cases, backup copies can be stored in a format in which the data is compressed, encrypted, deduplicated, and/or otherwise modified from the original native application format. For example, a backup copy may be stored in a compressed backup format that facilitates efficient long-term storage. Backup copies 116 can have relatively long retention periods as compared to primary data 112, which is generally highly changeable. Backup copies 116 may be stored on media with slower retrieval times than primary storage device 104. Some backup copies may have shorter retention periods than some other types of secondary copies 116, such as archive copies (described below). Backups may be stored at an offsite location.

Backup operations can include full backups, differential backups, incremental backups, “synthetic full” backups, and/or creating a “reference copy.” A full backup (or “standard full backup”) in some embodiments is generally a complete image of the data to be protected. However, because full backup copies can consume a relatively large amount of storage, it can be useful to use a full backup copy as a baseline and only store changes relative to the full backup copy afterwards.

A differential backup operation (or cumulative incremental backup operation) tracks and stores changes that occurred since the last full backup. Differential backups can grow quickly in size, but can restore relatively efficiently because a restore can be completed in some cases using only the full backup copy and the latest differential copy.

An incremental backup operation generally tracks and stores changes since the most recent backup copy of any type, which can greatly reduce storage utilization. In some cases, however, restoring can be lengthy compared to full or

26

differential backups because completing a restore operation may involve accessing a full backup in addition to multiple incremental backups.

Synthetic full backups generally consolidate data without directly backing up data from the client computing device. A synthetic full backup is created from the most recent full backup (i.e., standard or synthetic) and subsequent incremental and/or differential backups. The resulting synthetic full backup is identical to what would have been created had the last backup for the subclient been a standard full backup. Unlike standard full, incremental, and differential backups, however, a synthetic full backup does not actually transfer data from primary storage to the backup media, because it operates as a backup consolidator. A synthetic full backup extracts the index data of each participating subclient. Using this index data and the previously backed up user data images, it builds new full backup images (e.g., bitmaps), one for each subclient. The new backup images consolidate the index and user data stored in the related incremental, differential, and previous full backups into a synthetic backup file that fully represents the subclient (e.g., via pointers) but does not comprise all its constituent data.

Any of the above types of backup operations can be at the volume level, file level, or block level. Volume level backup operations generally involve copying of a data volume (e.g., a logical disk or partition) as a whole. In a file-level backup, information management system 100 generally tracks changes to individual files and includes copies of files in the backup copy. For block-level backups, files are broken into constituent blocks, and changes are tracked at the block level. Upon restore, system 100 reassembles the blocks into files in a transparent fashion. Far less data may actually be transferred and copied to secondary storage devices 108 during a file-level copy than a volume-level copy. Likewise, a block-level copy may transfer less data than a file-level copy, resulting in faster execution. However, restoring a relatively higher-granularity copy can result in longer restore times. For instance, when restoring a block-level copy, the process of locating and retrieving constituent blocks can sometimes take longer than restoring file-level backups.

A reference copy may comprise copy(ies) of selected objects from backed up data, typically to help organize data by keeping contextual information from multiple sources together, and/or help retain specific data for a longer period of time, such as for legal hold needs. A reference copy generally maintains data integrity, and when the data is restored, it may be viewed in the same format as the source data. In some embodiments, a reference copy is based on a specialized client, individual subclient and associated information management policies (e.g., storage policy, retention policy, etc.) that are administered within system 100.

Archive Operations

Because backup operations generally involve maintaining a version of the copied primary data 112 and also maintaining backup copies in secondary storage device(s) 108, they can consume significant storage capacity. To reduce storage consumption, an archive operation according to certain embodiments creates an archive copy 116 by both copying and removing source data. Or, seen another way, archive operations can involve moving some or all of the source data to the archive destination. Thus, data satisfying criteria for removal (e.g., data of a threshold age or size) may be removed from source storage. The source data may be primary data 112 or a secondary copy 116, depending on the situation. As with backup copies, archive copies can be stored in a format in which the data is compressed,

US 10,210,048 B2

27

encrypted, deduplicated, and/or otherwise modified from the format of the original application or source copy. In addition, archive copies may be retained for relatively long periods of time (e.g., years) and, in some cases are never deleted. In certain embodiments, archive copies may be made and kept for extended periods in order to meet compliance regulations.

Archiving can also serve the purpose of freeing up space in primary storage device(s) 104 and easing the demand on computational resources on client computing device 102. Similarly, when a secondary copy 116 is archived, the archive copy can therefore serve the purpose of freeing up space in the source secondary storage device(s) 108. Examples of data archiving operations are provided in U.S. Pat. No. 7,107,298.

Snapshot Operations

Snapshot operations can provide a relatively lightweight, efficient mechanism for protecting data. From an end-user viewpoint, a snapshot may be thought of as an “instant” image of primary data 112 at a given point in time, and may include state and/or status information relative to an application 110 that creates/manages primary data 112. In one embodiment, a snapshot may generally capture the directory structure of an object in primary data 112 such as a file or volume or other data set at a particular moment in time and may also preserve file attributes and contents. A snapshot in some cases is created relatively quickly, e.g., substantially instantly, using a minimum amount of file space, but may still function as a conventional file system backup.

A “hardware snapshot” (or “hardware-based snapshot”) operation occurs where a target storage device (e.g., a primary storage device 104 or a secondary storage device 108) performs the snapshot operation in a self-contained fashion, substantially independently, using hardware, firmware and/or software operating on the storage device itself. For instance, the storage device may perform snapshot operations generally without intervention or oversight from any of the other components of the system 100, e.g., a storage array may generate an “array-created” hardware snapshot and may also manage its storage, integrity, versioning, etc. In this manner, hardware snapshots can off-load other components of system 100 from snapshot processing. An array may receive a request from another component to take a snapshot and then proceed to execute the “hardware snapshot” operations autonomously, preferably reporting success to the requesting component.

A “software snapshot” (or “software-based snapshot”) operation, on the other hand, occurs where a component in system 100 (e.g., client computing device 102, etc.) implements a software layer that manages the snapshot operation via interaction with the target storage device. For instance, the component executing the snapshot management software layer may derive a set of pointers and/or data that represents the snapshot. The snapshot management software layer may then transmit the same to the target storage device, along with appropriate instructions for writing the snapshot. One example of a software snapshot product is Microsoft Volume Snapshot Service (VSS), which is part of the Microsoft Windows operating system.

Some types of snapshots do not actually create another physical copy of all the data as it existed at the particular point in time, but may simply create pointers that map files and directories to specific memory locations (e.g., to specific disk blocks) where the data resides as it existed at the particular point in time. For example, a snapshot copy may include a set of pointers derived from the file system or from an application. In some other cases, the snapshot may be

28

created at the block-level, such that creation of the snapshot occurs without awareness of the file system. Each pointer points to a respective stored data block, so that collectively, the set of pointers reflect the storage location and state of the data object (e.g., file(s) or volume(s) or data set(s)) at the point in time when the snapshot copy was created.

An initial snapshot may use only a small amount of disk space needed to record a mapping or other data structure representing or otherwise tracking the blocks that correspond to the current state of the file system. Additional disk space is usually required only when files and directories change later on. Furthermore, when files change, typically only the pointers which map to blocks are copied, not the blocks themselves. For example for “copy-on-write” snapshots, when a block changes in primary storage, the block is copied to secondary storage or cached in primary storage before the block is overwritten in primary storage, and the pointer to that block is changed to reflect the new location of that block. The snapshot mapping of file system data may also be updated to reflect the changed block(s) at that particular point in time. In some other cases, a snapshot includes a full physical copy of all or substantially all of the data represented by the snapshot. Further examples of snapshot operations are provided in U.S. Pat. No. 7,529,782. A snapshot copy in many cases can be made quickly and without significantly impacting primary computing resources because large amounts of data need not be copied or moved. In some embodiments, a snapshot may exist as a virtual file system, parallel to the actual file system. Users in some cases gain read-only access to the record of files and directories of the snapshot. By electing to restore primary data 112 from a snapshot taken at a given point in time, users may also return the current file system to the state of the file system that existed when the snapshot was taken.

Replication Operations

Replication is another type of secondary copy operation. Some types of secondary copies 116 periodically capture images of primary data 112 at particular points in time (e.g., backups, archives, and snapshots). However, it can also be useful for recovery purposes to protect primary data 112 in a more continuous fashion, by replicating primary data 112 substantially as changes occur. In some cases a replication copy can be a mirror copy, for instance, where changes made to primary data 112 are mirrored or substantially immediately copied to another location (e.g., to secondary storage device(s) 108). By copying each write operation to the replication copy, two storage systems are kept synchronized or substantially synchronized so that they are virtually identical at approximately the same time. Where entire disk volumes are mirrored, however, mirroring can require significant amount of storage space and utilizes a large amount of processing resources.

According to some embodiments, secondary copy operations are performed on replicated data that represents a recoverable state, or “known good state” of a particular application running on the source system. For instance, in certain embodiments, known good replication copies may be viewed as copies of primary data 112. This feature allows the system to directly access, copy, restore, back up, or otherwise manipulate the replication copies as if they were the “live” primary data 112. This can reduce access time, storage utilization, and impact on source applications 110, among other benefits. Based on known good state information, system 100 can replicate sections of application data that represent a recoverable state rather than rote copying of

US 10,210,048 B2

29

blocks of data. Examples of replication operations (e.g., continuous data replication) are provided in U.S. Pat. No. 7,617,262.

Deduplication/Single-Instancing Operations

Deduplication or single-instance storage is useful to reduce the amount of non-primary data. For instance, some or all of the above-described secondary copy operations can involve deduplication in some fashion. New data is read, broken down into data portions of a selected granularity (e.g., sub-file level blocks, files, etc.), compared with corresponding portions that are already in secondary storage, and only new/changed portions are stored. Portions that already exist are represented as pointers to the already-stored data. Thus, a deduplicated secondary copy **116** may comprise actual data portions copied from primary data **112** and may further comprise pointers to already-stored data, which is generally more storage-efficient than a full copy.

In order to streamline the comparison process, system **100** may calculate and/or store signatures (e.g., hashes or cryptographically unique IDs) corresponding to the individual source data portions and compare the signatures to already-stored data signatures, instead of comparing entire data portions. In some cases, only a single instance of each data portion is stored, and deduplication operations may therefore be referred to interchangeably as “single-instancing” operations. Depending on the implementation, however, deduplication operations can store more than one instance of certain data portions, yet still significantly reduce stored-data redundancy. Depending on the embodiment, deduplication portions such as data blocks can be of fixed or variable length. Using variable length blocks can enhance deduplication by responding to changes in the data stream, but can involve more complex processing. In some cases, system **100** utilizes a technique for dynamically aligning deduplication blocks based on changing content in the data stream, as described in U.S. Pat. No. 8,364,652.

System **100** can deduplicate in a variety of manners at a variety of locations. For instance, in some embodiments, system **100** implements “target-side” deduplication by deduplicating data at the media agent **144** after being received from data agent **142**. In some such cases, media agents **144** are generally configured to manage the deduplication process. For instance, one or more of the media agents **144** maintain a corresponding deduplication database that stores deduplication information (e.g., datablock signatures). Examples of such a configuration are provided in U.S. Pat. No. 9,020,900. Instead of or in combination with “target-side” deduplication, “source-side” (or “client-side”) deduplication can also be performed, e.g., to reduce the amount of data to be transmitted by data agent **142** to media agent **144**. Storage manager **140** may communicate with other components within system **100** via network protocols and cloud service provider APIs to facilitate cloud-based deduplication/single instancing, as exemplified in U.S. Pat. No. 8,954,446. Some other deduplication/single instancing techniques are described in U.S. Pat. Pub. No. 2006/0224846 and in U.S. Pat. No. 9,098,495.

Information Lifecycle Management and Hierarchical Storage Management

In some embodiments, files and other data over their lifetime move from more expensive quick-access storage to less expensive slower-access storage. Operations associated with moving data through various tiers of storage are sometimes referred to as information lifecycle management (ILM) operations.

One type of ILM operation is a hierarchical storage management (HSM) operation, which generally automati-

30

cally moves data between classes of storage devices, such as from high-cost to low-cost storage devices. For instance, an HSM operation may involve movement of data from primary storage devices **104** to secondary storage devices **108**, or between tiers of secondary storage devices **108**. With each tier, the storage devices may be progressively cheaper, have relatively slower access/restore times, etc. For example, movement of data between tiers may occur as data becomes less important over time. In some embodiments, an HSM operation is similar to archiving in that creating an HSM copy may (though not always) involve deleting some of the source data, e.g., according to one or more criteria related to the source data. For example, an HSM copy may include primary data **112** or a secondary copy **116** that exceeds a given size threshold or a given age threshold. Often, and unlike some types of archive copies, HSM data that is removed or aged from the source is replaced by a logical reference pointer or stub. The reference pointer or stub can be stored in the primary storage device **104** or other source storage device, such as a secondary storage device **108** to replace the deleted source data and to point to or otherwise indicate the new location in (another) secondary storage device **108**.

For example, files are generally moved between higher and lower cost storage depending on how often the files are accessed. When a user requests access to HSM data that has been removed or migrated, system **100** uses the stub to locate the data and may make recovery of the data appear transparent, even though the HSM data may be stored at a location different from other source data. In this manner, the data appears to the user (e.g., in file system browsing windows and the like) as if it still resides in the source location (e.g., in a primary storage device **104**). The stub may include metadata associated with the corresponding data, so that a file system and/or application can provide some information about the data object and/or a limited-functionality version (e.g., a preview) of the data object.

An HSM copy may be stored in a format other than the native application format (e.g., compressed, encrypted, deduplicated, and/or otherwise modified). In some cases, copies which involve the removal of data from source storage and the maintenance of stub or other logical reference information on source storage may be referred to generally as “on-line archive copies.” On the other hand, copies which involve the removal of data from source storage without the maintenance of stub or other logical reference information on source storage may be referred to as “off-line archive copies.” Examples of HSM and ILM techniques are provided in U.S. Pat. No. 7,343,453.

Auxiliary Copy Operations

An auxiliary copy is generally a copy of an existing secondary copy **116**. For instance, an initial secondary copy **116** may be derived from primary data **112** or from data residing in secondary storage subsystem **118**, whereas an auxiliary copy is generated from the initial secondary copy **116**. Auxiliary copies provide additional standby copies of data and may reside on different secondary storage devices **108** than the initial secondary copies **116**. Thus, auxiliary copies can be used for recovery purposes if initial secondary copies **116** become unavailable. Exemplary auxiliary copy techniques are described in further detail in U.S. Pat. No. 8,230,195.

Disaster-Recovery Copy Operations

System **100** may also make and retain disaster recovery copies, often as secondary, high-availability disk copies. System **100** may create secondary copies and store them at disaster recovery locations using auxiliary copy or replica-

US 10,210,048 B2

31

tion operations, such as continuous data replication technologies. Depending on the particular data protection goals, disaster recovery locations can be remote from the client computing devices 102 and primary storage devices 104, remote from some or all of the secondary storage devices 108, or both.

Data Manipulation, Including Encryption and Compression

Data manipulation and processing may include encryption and compression as well as integrity marking and checking, formatting for transmission, formatting for storage, etc. Data may be manipulated “client-side” by data agent 142 as well as “target-side” by media agent 144 in the course of creating secondary copy 116, or conversely in the course of restoring data from secondary to primary.

Encryption Operations

System 100 in some cases is configured to process data (e.g., files or other data objects, primary data 112, secondary copies 116, etc.), according to an appropriate encryption algorithm (e.g., Blowfish, Advanced Encryption Standard (AES), Triple Data Encryption Standard (3-DES), etc.) to limit access and provide data security. System 100 in some cases encrypts the data at the client level, such that client computing devices 102 (e.g., data agents 142) encrypt the data prior to transferring it to other components, e.g., before sending the data to media agents 144 during a secondary copy operation. In such cases, client computing device 102 may maintain or have access to an encryption key or passphrase for decrypting the data upon restore. Encryption can also occur when media agent 144 creates auxiliary copies or archive copies. Encryption may be applied in creating a secondary copy 116 of a previously unencrypted secondary copy 116, without limitation. In further embodiments, secondary storage devices 108 can implement built-in, high performance hardware-based encryption.

Compression Operations

Similar to encryption, system 100 may also or alternatively compress data in the course of generating a secondary copy 116. Compression encodes information such that fewer bits are needed to represent the information as compared to the original representation. Compression techniques are well known in the art. Compression operations may apply one or more data compression algorithms. Compression may be applied in creating a secondary copy 116 of a previously uncompressed secondary copy, e.g., when making archive copies or disaster recovery copies. The use of compression may result in metadata that specifies the nature of the compression, so that data may be uncompressed on restore if appropriate.

Data Analysis, Reporting, and Management Operations

Data analysis, reporting, and management operations can differ from data movement operations in that they do not necessarily involve copying, migration or other transfer of data between different locations in the system. For instance, data analysis operations may involve processing (e.g., offline processing) or modification of already stored primary data 112 and/or secondary copies 116. However, in some embodiments data analysis operations are performed in conjunction with data movement operations. Some data analysis operations include content indexing operations and classification operations which can be useful in leveraging data under management to enhance search and other features.

Classification Operations/Content Indexing

In some embodiments, information management system 100 analyzes and indexes characteristics, content, and metadata associated with primary data 112 (“online content

32

indexing”) and/or secondary copies 116 (“off-line content indexing”). Content indexing can identify files or other data objects based on content (e.g., user-defined keywords or phrases, other keywords/phrases that are not defined by a user, etc.), and/or metadata (e.g., email metadata such as “to,” “from,” “cc,” “bcc,” attachment name, received time, etc.). Content indexes may be searched and search results may be restored.

System 100 generally organizes and catalogues the results into a content index, which may be stored within media agent database 152, for example. The content index can also include the storage locations of or pointer references to indexed data in primary data 112 and/or secondary copies 116. Results may also be stored elsewhere in system 100 (e.g., in primary storage device 104 or in secondary storage device 108). Such content index data provides storage manager 140 or other components with an efficient mechanism for locating primary data 112 and/or secondary copies 116 of data objects that match particular criteria, thus greatly increasing the search speed capability of system 100. For instance, search criteria can be specified by a user through user interface 158 of storage manager 140. Moreover, when system 100 analyzes data and/or metadata in secondary copies 116 to create an “off-line content index,” this operation has no significant impact on the performance of client computing devices 102 and thus does not take a toll on the production environment. Examples of content indexing techniques are provided in U.S. Pat. No. 8,170,995.

One or more components, such as a content index engine, can be configured to scan data and/or associated metadata for classification purposes to populate a database (or other data structure) of information, which can be referred to as a “data classification database” or a “metabase.” Depending on the embodiment, the data classification database(s) can be organized in a variety of different ways, including centralization, logical sub-divisions, and/or physical sub-divisions. For instance, one or more data classification databases may be associated with different subsystems or tiers within system 100. As an example, there may be a first metabase associated with primary storage subsystem 117 and a second metabase associated with secondary storage subsystem 118. In other cases, metabase(s) may be associated with individual components, e.g., client computing devices 102 and/or media agents 144. In some embodiments, a data classification database may reside as one or more data structures within management database 146, may be otherwise associated with storage manager 140, and/or may reside as a separate component. In some cases, metabase(s) may be included in separate database(s) and/or on separate storage device(s) from primary data 112 and/or secondary copies 116, such that operations related to the metabase(s) do not significantly impact performance on other components of system 100. In other cases, metabase(s) may be stored along with primary data 112 and/or secondary copies 116. Files or other data objects can be associated with identifiers (e.g., tag entries, etc.) to facilitate searches of stored data objects. Among a number of other benefits, the metabase can also allow efficient, automatic identification of files or other data objects to associate with secondary copy or other information management operations. For instance, a metabase can dramatically improve the speed with which system 100 can search through and identify data as compared to other approaches that involve scanning an entire file system. Examples of metabases and data classification operations are provided in U.S. Pat. Nos. 7,734,669 and 7,747,579.

US 10,210,048 B2

33

Management and Reporting Operations

Certain embodiments leverage the integrated ubiquitous nature of system **100** to provide useful system-wide management and reporting. Operations management can generally include monitoring and managing the health and performance of system **100** by, without limitation, performing error tracking, generating granular storage/performance metrics (e.g., job success/failure information, deduplication efficiency, etc.), generating storage modeling and costing information, and the like. As an example, storage manager **140** or another component in system **100** may analyze traffic patterns and suggest and/or automatically route data to minimize congestion. In some embodiments, the system can generate predictions relating to storage operations or storage operation information. Such predictions, which may be based on a trending analysis, may predict various network operations or resource usage, such as network traffic levels, storage media use, use of bandwidth of communication links, use of media agent components, etc. Further examples of traffic analysis, trend analysis, prediction generation, and the like are described in U.S. Pat. No. 7,343,453.

In some configurations having a hierarchy of storage operation cells, a master storage manager **140** may track the status of subordinate cells, such as the status of jobs, system components, system resources, and other items, by communicating with storage managers **140** (or other components) in the respective storage operation cells. Moreover, the master storage manager **140** may also track status by receiving periodic status updates from the storage managers **140** (or other components) in the respective cells regarding jobs, system components, system resources, and other items. In some embodiments, a master storage manager **140** may store status information and other information regarding its associated storage operation cells and other system information in its management database **146** and/or index **150** (or in another location). The master storage manager **140** or other component may also determine whether certain storage-related or other criteria are satisfied, and may perform an action or trigger event (e.g., data migration) in response to the criteria being satisfied, such as where a storage threshold is met for a particular volume, or where inadequate protection exists for certain data. For instance, data from one or more storage operation cells is used to dynamically and automatically mitigate recognized risks, and/or to advise users of risks or suggest actions to mitigate these risks. For example, an information management policy may specify certain requirements (e.g., that a storage device should maintain a certain amount of free space, that secondary copies should occur at a particular interval, that data should be aged and migrated to other storage after a particular period, that data on a secondary volume should always have a certain level of availability and be restorable within a given time period, that data on a secondary volume may be mirrored or otherwise migrated to a specified number of other volumes, etc.). If a risk condition or other criterion is triggered, the system may notify the user of these conditions and may suggest (or automatically implement) a mitigation action to address the risk. For example, the system may indicate that data from a primary copy **112** should be migrated to a secondary storage device **108** to free up space on primary storage device **104**. Examples of the use of risk factors and other triggering criteria are described in U.S. Pat. No. 7,343,453.

In some embodiments, system **100** may also determine whether a metric or other indication satisfies particular storage criteria sufficient to perform an action. For example, a storage policy or other definition might indicate that a

34

storage manager **140** should initiate a particular action if a storage metric or other indication drops below or otherwise fails to satisfy specified criteria such as a threshold of data protection. In some embodiments, risk factors may be quantified into certain measurable service or risk levels. For example, certain applications and associated data may be considered to be more important relative to other data and services. Financial compliance data, for example, may be of greater importance than marketing materials, etc. Network administrators may assign priority values or “weights” to certain data and/or applications corresponding to the relative importance. The level of compliance of secondary copy operations specified for these applications may also be assigned a certain value. Thus, the health, impact, and overall importance of a service may be determined, such as by measuring the compliance value and calculating the product of the priority value and the compliance value to determine the “service level” and comparing it to certain operational thresholds to determine whether it is acceptable. Further examples of the service level determination are provided in U.S. Pat. No. 7,343,453.

System **100** may additionally calculate data costing and data availability associated with information management operation cells. For instance, data received from a cell may be used in conjunction with hardware-related information and other information about system elements to determine the cost of storage and/or the availability of particular data. Exemplary information generated could include how fast a particular department is using up available storage space, how long data would take to recover over a particular pathway from a particular secondary storage device, costs over time, etc. Moreover, in some embodiments, such information may be used to determine or predict the overall cost associated with the storage of certain information. The cost associated with hosting a certain application may be based, at least in part, on the type of media on which the data resides, for example. Storage devices may be assigned to a particular cost categories, for example. Further examples of costing techniques are described in U.S. Pat. No. 7,343,453.

Any of the above types of information (e.g., information related to trending, predictions, job, cell or component status, risk, service level, costing, etc.) can generally be provided to users via user interface **158** in a single integrated view or console (not shown). Report types may include: scheduling, event management, media management and data aging. Available reports may also include backup history, data aging history, auxiliary copy history, job history, library and drive, media in library, restore history, and storage policy, etc., without limitation. Such reports may be specified and created at a certain point in time as a system analysis, forecasting, or provisioning tool. Integrated reports may also be generated that illustrate storage and performance metrics, risks and storage costing information. Moreover, users may create their own reports based on specific needs. User interface **158** can include an option to graphically depict the various components in the system using appropriate icons. As one example, user interface **158** may provide a graphical depiction of primary storage devices **104**, secondary storage devices **108**, data agents **142** and/or media agents **144**, and their relationship to one another in system **100**.

In general, the operations management functionality of system **100** can facilitate planning and decision-making. For example, in some embodiments, a user may view the status of some or all jobs as well as the status of each component of information management system **100**. Users may then plan and make decisions based on this data. For instance, a

US 10,210,048 B2

35

user may view high-level information regarding secondary copy operations for system **100**, such as job status, component status, resource status (e.g., communication pathways, etc.), and other information. The user may also drill down or use other means to obtain more detailed information regarding a particular component, job, or the like. Further examples are provided in U.S. Pat. No. 7,343,453.

System **100** can also be configured to perform system-wide e-discovery operations in some embodiments. In general, e-discovery operations provide a unified collection and search capability for data in the system, such as data stored in secondary storage devices **108** (e.g., backups, archives, or other secondary copies **116**). For example, system **100** may construct and maintain a virtual repository for data stored in system **100** that is integrated across source applications **110**, different storage device types, etc. According to some embodiments, e-discovery utilizes other techniques described herein, such as data classification and/or content indexing.

Information Management Policies

An information management policy **148** can include a data structure or other information source that specifies a set of parameters (e.g., criteria and rules) associated with secondary copy and/or other information management operations.

One type of information management policy **148** is a “storage policy.” According to certain embodiments, a storage policy generally comprises a data structure or other information source that defines (or includes information sufficient to determine) a set of preferences or other criteria for performing information management operations. Storage policies can include one or more of the following: (1) what data will be associated with the storage policy, e.g., subclient; (2) a destination to which the data will be stored; (3) datapath information specifying how the data will be communicated to the destination; (4) the type of secondary copy operation to be performed; and (5) retention information specifying how long the data will be retained at the destination (see, e.g., FIG. 1E). Data associated with a storage policy can be logically organized into subclients, which may represent primary data **112** and/or secondary copies **116**. A subclient may represent static or dynamic associations of portions of a data volume. Subclients may represent mutually exclusive portions. Thus, in certain embodiments, a portion of data may be given a label and the association is stored as a static entity in an index, database or other storage location. Subclients may also be used as an effective administrative scheme of organizing data according to data type, department within the enterprise, storage preferences, or the like. Depending on the configuration, subclients can correspond to files, folders, virtual machines, databases, etc. In one exemplary scenario, an administrator may find it preferable to separate e-mail data from financial data using two different subclients.

A storage policy can define where data is stored by specifying a target or destination storage device (or group of storage devices). For instance, where the secondary storage device **108** includes a group of disk libraries, the storage policy may specify a particular disk library for storing the subclients associated with the policy. As another example, where the secondary storage devices **108** include one or more tape libraries, the storage policy may specify a particular tape library for storing the subclients associated with the storage policy, and may also specify a drive pool and a tape pool defining a group of tape drives and a group of tapes, respectively, for use in storing the subclient data. While information in the storage policy can be statically

36

assigned in some cases, some or all of the information in the storage policy can also be dynamically determined based on criteria set forth in the storage policy. For instance, based on such criteria, a particular destination storage device(s) or other parameter of the storage policy may be determined based on characteristics associated with the data involved in a particular secondary copy operation, device availability (e.g., availability of a secondary storage device **108** or a media agent **144**), network status and conditions (e.g., identified bottlenecks), user credentials, and the like.

Datapath information can also be included in the storage policy. For instance, the storage policy may specify network pathways and components to utilize when moving the data to the destination storage device(s). In some embodiments, the storage policy specifies one or more media agents **144** for conveying data associated with the storage policy between the source and destination. A storage policy can also specify the type(s) of associated operations, such as backup, archive, snapshot, auxiliary copy, or the like. Furthermore, retention parameters can specify how long the resulting secondary copies **116** will be kept (e.g., a number of days, months, years, etc.), perhaps depending on organizational needs and/or compliance criteria.

When adding a new client computing device **102**, administrators can manually configure information management policies **148** and/or other settings, e.g., via user interface **158**. However, this can be an involved process resulting in delays, and it may be desirable to begin data protection operations quickly, without awaiting human intervention. Thus, in some embodiments, system **100** automatically applies a default configuration to client computing device **102**. As one example, when one or more data agent(s) **142** are installed on a client computing device **102**, the installation script may register the client computing device **102** with storage manager **140**, which in turn applies the default configuration to the new client computing device **102**. In this manner, data protection operations can begin substantially immediately. The default configuration can include a default storage policy, for example, and can specify any appropriate information sufficient to begin data protection operations. This can include a type of data protection operation, scheduling information, a target secondary storage device **108**, data path information (e.g., a particular media agent **144**), and the like.

Another type of information management policy **148** is a “scheduling policy,” which specifies when and how often to perform operations. Scheduling parameters may specify with what frequency (e.g., hourly, weekly, daily, event-based, etc.) or under what triggering conditions secondary copy or other information management operations are to take place. Scheduling policies in some cases are associated with particular components, such as a subclient, client computing device **102**, and the like.

Another type of information management policy **148** is an “audit policy” (or “security policy”), which comprises preferences, rules and/or criteria that protect sensitive data in system **100**. For example, an audit policy may define “sensitive objects” which are files or data objects that contain particular keywords (e.g., “confidential,” or “privileged”) and/or are associated with particular keywords (e.g., in metadata) or particular flags (e.g., in metadata identifying a document or email as personal, confidential, etc.). An audit policy may further specify rules for handling sensitive objects. As an example, an audit policy may require that a reviewer approve the transfer of any sensitive objects to a cloud storage site, and that if approval is denied for a particular sensitive object, the sensitive object should be

US 10,210,048 B2

37

transferred to a local primary storage device **104** instead. To facilitate this approval, the audit policy may further specify how a secondary storage computing device **106** or other system component should notify a reviewer that a sensitive object is slated for transfer.

Another type of information management policy **148** is a “provisioning policy,” which can include preferences, priorities, rules, and/or criteria that specify how client computing devices **102** (or groups thereof) may utilize system resources, such as available storage on cloud storage and/or network bandwidth. A provisioning policy specifies, for example, data quotas for particular client computing devices **102** (e.g., a number of gigabytes that can be stored monthly, quarterly or annually). Storage manager **140** or other components may enforce the provisioning policy. For instance, media agents **144** may enforce the policy when transferring data to secondary storage devices **108**. If a client computing device **102** exceeds a quota, a budget for the client computing device **102** (or associated department) may be adjusted accordingly or an alert may trigger.

While the above types of information management policies **148** are described as separate policies, one or more of these can be generally combined into a single information management policy **148**. For instance, a storage policy may also include or otherwise be associated with one or more scheduling, audit, or provisioning policies or operational parameters thereof. Moreover, while storage policies are typically associated with moving and storing data, other policies may be associated with other types of information management operations. The following is a non-exhaustive list of items that information management policies **148** may specify:

- schedules or other timing information, e.g., specifying when and/or how often to perform information management operations;
- the type of secondary copy **116** and/or copy format (e.g., snapshot, backup, archive, HSM, etc.);
- a location or a class or quality of storage for storing secondary copies **116** (e.g., one or more particular secondary storage devices **108**);
- preferences regarding whether and how to encrypt, compress, deduplicate, or otherwise modify or transform secondary copies **116**;
- which system components and/or network pathways (e.g., preferred media agents **144**) should be used to perform secondary storage operations;
- resource allocation among different computing devices or other system components used in performing information management operations (e.g., bandwidth allocation, available storage capacity, etc.);
- whether and how to synchronize or otherwise distribute files or other data objects across multiple computing devices or hosted services; and
- retention information specifying the length of time primary data **112** and/or secondary copies **116** should be retained, e.g., in a particular class or tier of storage devices, or within the system **100**.

Information management policies **148** can additionally specify or depend on historical or current criteria that may be used to determine which rules to apply to a particular data object, system component, or information management operation, such as:

- frequency with which primary data **112** or a secondary copy **116** of a data object or metadata has been or is predicted to be used, accessed, or modified;
- time-related factors (e.g., aging information such as time since the creation or modification of a data object);

38

deduplication information (e.g., hashes, data blocks, deduplication block size, deduplication efficiency or other metrics);

an estimated or historic usage or cost associated with different components (e.g., with secondary storage devices **108**);

the identity of users, applications **110**, client computing devices **102** and/or other computing devices that created, accessed, modified, or otherwise utilized primary data **112** or secondary copies **116**;

a relative sensitivity (e.g., confidentiality, importance) of a data object, e.g., as determined by its content and/or metadata;

the current or historical storage capacity of various storage devices;

the current or historical network capacity of network pathways connecting various components within the storage operation cell;

access control lists or other security information; and

the content of a particular data object (e.g., its textual content) or of metadata associated with the data object.

Exemplary Storage Policy and Secondary Copy Operations

FIG. 1E includes a data flow diagram depicting performance of secondary copy operations by an embodiment of information management system **100**, according to an exemplary storage policy **148A**. System **100** includes a storage manager **140**, a client computing device **102** having a file system data agent **142A** and an email data agent **142B** operating thereon, a primary storage device **104**, two media agents **144A**, **144B**, and two secondary storage devices **108**: a disk library **108A** and a tape library **108B**. As shown, primary storage device **104** includes primary data **112A**, which is associated with a logical grouping of data associated with a file system (“file system subclient”), and primary data **112B**, which is a logical grouping of data associated with email (“email subclient”). The techniques described with respect to FIG. 1E can be utilized in conjunction with data that is otherwise organized as well.

As indicated by the dashed box, the second media agent **144B** and tape library **108B** are “off-site,” and may be remotely located from the other components in system **100** (e.g., in a different city, office building, etc.). Indeed, “off-site” may refer to a magnetic tape located in remote storage, which must be manually retrieved and loaded into a tape drive to be read. In this manner, information stored on the tape library **108B** may provide protection in the event of a disaster or other failure at the main site(s) where data is stored.

The file system subclient **112A** in certain embodiments generally comprises information generated by the file system and/or operating system of client computing device **102**, and can include, for example, file system data (e.g., regular files, file tables, mount points, etc.), operating system data (e.g., registries, event logs, etc.), and the like. The e-mail subclient **112B** can include data generated by an e-mail application operating on client computing device **102**, e.g., mailbox information, folder information, emails, attachments, associated database information, and the like. As described above, the subclients can be logical containers, and the data included in the corresponding primary data **112A** and **112B** may or may not be stored contiguously.

The exemplary storage policy **148A** includes backup copy preferences or rule set **160**, disaster recovery copy preferences or rule set **162**, and compliance copy preferences or rule set **164**. Backup copy rule set **160** specifies that it is associated with file system subclient **166** and email subclient

US 10,210,048 B2

39

168. Each of subclients 166 and 168 are associated with the particular client computing device 102. Backup copy rule set 160 further specifies that the backup operation will be written to disk library 108A and designates a particular media agent 144A to convey the data to disk library 108A. Finally, backup copy rule set 160 specifies that backup copies created according to rule set 160 are scheduled to be generated hourly and are to be retained for 30 days. In some other embodiments, scheduling information is not included in storage policy 148A and is instead specified by a separate scheduling policy.

Disaster recovery copy rule set 162 is associated with the same two subclients 166 and 168. However, disaster recovery copy rule set 162 is associated with tape library 108B, unlike backup copy rule set 160. Moreover, disaster recovery copy rule set 162 specifies that a different media agent, namely 144B, will convey data to tape library 108B. Disaster recovery copies created according to rule set 162 will be retained for 60 days and will be generated daily. Disaster recovery copies generated according to disaster recovery copy rule set 162 can provide protection in the event of a disaster or other catastrophic data loss that would affect the backup copy 116A maintained on disk library 108A.

Compliance copy rule set 164 is only associated with the email subclient 168, and not the file system subclient 166. Compliance copies generated according to compliance copy rule set 164 will therefore not include primary data 112A from the file system subclient 166. For instance, the organization may be under an obligation to store and maintain copies of email data for a particular period of time (e.g., 10 years) to comply with state or federal regulations, while similar regulations do not apply to file system data. Compliance copy rule set 164 is associated with the same tape library 108B and media agent 144B as disaster recovery copy rule set 162, although a different storage device or media agent could be used in other embodiments. Finally, compliance copy rule set 164 specifies that the copies it governs will be generated quarterly and retained for 10 years.

Secondary Copy Jobs

A logical grouping of secondary copy operations governed by a rule set and being initiated at a point in time may be referred to as a "secondary copy job" (and sometimes may be called a "backup job," even though it is not necessarily limited to creating only backup copies). Secondary copy jobs may be initiated on demand as well. Steps 1-9 below illustrate three secondary copy jobs based on storage policy 148A.

Referring to FIG. 1E, at step 1, storage manager 140 initiates a backup job according to the backup copy rule set 160, which logically comprises all the secondary copy operations necessary to effectuate rules 160 in storage policy 148A every hour, including steps 1-4 occurring hourly. For instance, a scheduling service running on storage manager 140 accesses backup copy rule set 160 or a separate scheduling policy associated with client computing device 102 and initiates a backup job on an hourly basis. Thus, at the scheduled time, storage manager 140 sends instructions to client computing device 102 (i.e., to both data agent 142A and data agent 142B) to begin the backup job.

At step 2, file system data agent 142A and email data agent 142B on client computing device 102 respond to instructions from storage manager 140 by accessing and processing the respective subclient primary data 112A and 112B involved in the backup copy operation, which can be found in primary storage device 104. Because the secondary copy operation is a backup copy operation, the data agent(s)

40

142A, 142B may format the data into a backup format or otherwise process the data suitable for a backup copy.

At step 3, client computing device 102 communicates the processed file system data (e.g., using file system data agent 142A) and the processed email data (e.g., using email data agent 142B) to the first media agent 144A according to backup copy rule set 160, as directed by storage manager 140. Storage manager 140 may further keep a record in management database 146 of the association between media agent 144A and one or more of: client computing device 102, file system subclient 112A, file system data agent 142A, email subclient 112B, email data agent 142B, and/or backup copy 116A.

The target media agent 144A receives the data-agent-processed data from client computing device 102, and at step 4 generates and conveys backup copy 116A to disk library 108A to be stored as backup copy 116A, again at the direction of storage manager 140 and according to backup copy rule set 160. Media agent 144A can also update its index 153 to include data and/or metadata related to backup copy 116A, such as information indicating where the backup copy 116A resides on disk library 108A, where the email copy resides, where the file system copy resides, data and metadata for cache retrieval, etc. Storage manager 140 may similarly update its index 150 to include information relating to the secondary copy operation, such as information relating to the type of operation, a physical location associated with one or more copies created by the operation, the time the operation was performed, status information relating to the operation, the components involved in the operation, and the like. In some cases, storage manager 140 may update its index 150 to include some or all of the information stored in index 153 of media agent 144A. At this point, the backup job may be considered complete. After the 30-day retention period expires, storage manager 140 instructs media agent 144A to delete backup copy 116A from disk library 108A and indexes 150 and/or 153 are updated accordingly.

At step 5, storage manager 140 initiates another backup job for a disaster recovery copy according to the disaster recovery rule set 162. Illustratively this includes steps 5-7 occurring daily for creating disaster recovery copy 1168. Illustratively, and by way of illustrating the scalable aspects and off-loading principles embedded in system 100, disaster recovery copy 1168 is based on backup copy 116A and not on primary data 112A and 112B.

At step 6, illustratively based on instructions received from storage manager 140 at step 5, the specified media agent 144B retrieves the most recent backup copy 116A from disk library 108A.

At step 7, again at the direction of storage manager 140 and as specified in disaster recovery copy rule set 162, media agent 144B uses the retrieved data to create a disaster recovery copy 1168 and store it to tape library 108B. In some cases, disaster recovery copy 1168 is a direct, mirror copy of backup copy 116A, and remains in the backup format. In other embodiments, disaster recovery copy 1168 may be further compressed or encrypted, or may be generated in some other manner, such as by using primary data 112A and 112B from primary storage device 104 as sources. The disaster recovery copy operation is initiated once a day and disaster recovery copies 1168 are deleted after 60 days; indexes 153 and/or 150 are updated accordingly when/after each information management operation is executed and/or completed. The present backup job may be considered completed.

At step 8, storage manager 140 initiates another backup job according to compliance rule set 164, which performs

US 10,210,048 B2

41

steps 8-9 quarterly to create compliance copy 116C. For instance, storage manager 140 instructs media agent 144B to create compliance copy 116C on tape library 108B, as specified in the compliance copy rule set 164.

At step 9 in the example, compliance copy 116C is generated using disaster recovery copy 116B as the source. This is efficient, because disaster recovery copy resides on the same secondary storage device and thus no network resources are required to move the data. In other embodiments, compliance copy 116C is instead generated using primary data 112B corresponding to the email subclient or using backup copy 116A from disk library 108A as source data. As specified in the illustrated example, compliance copies 116C are created quarterly, and are deleted after ten years, and indexes 153 and/or 150 are kept up-to-date accordingly.

Exemplary Applications of Storage Policies—Information Governance Policies and Classification

Again referring to FIG. 1E, storage manager 140 may permit a user to specify aspects of storage policy 148A. For example, the storage policy can be modified to include information governance policies to define how data should be managed in order to comply with a certain regulation or business objective. The various policies may be stored, for example, in management database 146. An information governance policy may align with one or more compliance tasks that are imposed by regulations or business requirements. Examples of information governance policies might include a Sarbanes-Oxley policy, a HIPAA policy, an electronic discovery (e-discovery) policy, and so on.

Information governance policies allow administrators to obtain different perspectives on an organization's online and offline data, without the need for a dedicated data silo created solely for each different viewpoint. As described previously, the data storage systems herein build an index that reflects the contents of a distributed data set that spans numerous clients and storage devices, including both primary data and secondary copies, and online and offline copies. An organization may apply multiple information governance policies in a top-down manner over that unified data set and indexing schema in order to view and manipulate the data set through different lenses, each of which is adapted to a particular compliance or business goal. Thus, for example, by applying an e-discovery policy and a Sarbanes-Oxley policy, two different groups of users in an organization can conduct two very different analyses of the same underlying physical set of data/copies, which may be distributed throughout the information management system.

An information governance policy may comprise a classification policy, which defines a taxonomy of classification terms or tags relevant to a compliance task and/or business objective. A classification policy may also associate a defined tag with a classification rule. A classification rule defines a particular combination of criteria, such as users who have created, accessed or modified a document or data object; file or application types; content or metadata keywords; clients or storage locations; dates of data creation and/or access; review status or other status within a workflow (e.g., reviewed or un-reviewed); modification times or types of modifications; and/or any other data attributes in any combination, without limitation. A classification rule may also be defined using other classification tags in the taxonomy. The various criteria used to define a classification rule may be combined in any suitable fashion, for example, via Boolean operators, to define a complex classification rule. As an example, an e-discovery classification policy might define a classification tag "privileged" that is associ-

42

ated with documents or data objects that (1) were created or modified by legal department staff, or (2) were sent to or received from outside counsel via email, or (3) contain one of the following keywords: "privileged" or "attorney" or "counsel," or other like terms. Accordingly, all these documents or data objects will be classified as "privileged."

One specific type of classification tag, which may be added to an index at the time of indexing, is an "entity tag." An entity tag may be, for example, any content that matches a defined data mask format. Examples of entity tags might include, e.g., social security numbers (e.g., any numerical content matching the formatting mask XXX-XX-XXXX), credit card numbers (e.g., content having a 13-16 digit string of numbers), SKU numbers, product numbers, etc. A user may define a classification policy by indicating criteria, parameters or descriptors of the policy via a graphical user interface, such as a form or page with fields to be filled in, pull-down menus or entries allowing one or more of several options to be selected, buttons, sliders, hypertext links or other known user interface tools for receiving user input, etc. For example, a user may define certain entity tags, such as a particular product number or project ID. In some implementations, the classification policy can be implemented using cloud-based techniques. For example, the storage devices may be cloud storage devices, and the storage manager 140 may execute cloud service provider API over a network to classify data stored on cloud storage devices. Restore Operations from Secondary Copies

While not shown in FIG. 1E, at some later point in time, a restore operation can be initiated involving one or more of secondary copies 116A, 116B, and 116C. A restore operation logically takes a selected secondary copy 116, reverses the effects of the secondary copy operation that created it, and stores the restored data to primary storage where a client computing device 102 may properly access it as primary data. A media agent 144 and an appropriate data agent 142 (e.g., executing on the client computing device 102) perform the tasks needed to complete a restore operation. For example, data that was encrypted, compressed, and/or deduplicated in the creation of secondary copy 116 will be correspondingly rehydrated (reversing deduplication), uncompressed, and unencrypted into a format appropriate to primary data. Metadata stored within or associated with the secondary copy 116 may be used during the restore operation. In general, restored data should be indistinguishable from other primary data 112. Preferably, the restored data has fully regained the native format that may make it immediately usable by application 110.

As one example, a user may manually initiate a restore of backup copy 116A, e.g., by interacting with user interface 158 of storage manager 140 or with a web-based console with access to system 100. Storage manager 140 may access data in its index 150 and/or management database 146 (and/or the respective storage policy 148A) associated with the selected backup copy 116A to identify the appropriate media agent 144A and/or secondary storage device 108A where the secondary copy resides. The user may be presented with a representation (e.g., stub, thumbnail, listing, etc.) and metadata about the selected secondary copy, in order to determine whether this is the appropriate copy to be restored, e.g., date that the original primary data was created. Storage manager 140 will then instruct media agent 144A and an appropriate data agent 142 on the target client computing device 102 to restore secondary copy 116A to primary storage device 104. A media agent may be selected for use in the restore operation based on a load balancing algorithm, an availability based algorithm, or other criteria.

US 10,210,048 B2

43

The selected media agent, e.g., **144A**, retrieves secondary copy **116A** from disk library **108A**. For instance, media agent **144A** may access its index **153** to identify a location of backup copy **116A** on disk library **108A**, or may access location information residing on disk library **108A** itself.

In some cases a backup copy **116A** that was recently created or accessed, may be cached to speed up the restore operation. In such a case, media agent **144A** accesses a cached version of backup copy **116A** residing in index **153**, without having to access disk library **108A** for some or all of the data. Once it has retrieved backup copy **116A**, the media agent **144A** communicates the data to the requesting client computing device **102**. Upon receipt, file system data agent **142A** and email data agent **142B** may unpack (e.g., restore from a backup format to the native application format) the data in backup copy **116A** and restore the unpackaged data to primary storage device **104**. In general, secondary copies **116** may be restored to the same volume or folder in primary storage device **104** from which the secondary copy was derived; to another storage location or client computing device **102**; to shared storage, etc. In some cases, the data may be restored so that it may be used by an application **110** of a different version/vintage from the application that created the original primary data **112**.

Exemplary Secondary Copy Formatting

The formatting and structure of secondary copies **116** can vary depending on the embodiment. In some cases, secondary copies **116** are formatted as a series of logical data units or “chunks” (e.g., 512 MB, 1 GB, 2 GB, 4 GB, or 8 GB chunks). This can facilitate efficient communication and writing to secondary storage devices **108**, e.g., according to resource availability. For example, a single secondary copy **116** may be written on a chunk-by-chunk basis to one or more secondary storage devices **108**. In some cases, users can select different chunk sizes, e.g., to improve throughput to tape storage devices. Generally, each chunk can include a header and a payload. The payload can include files (or other data units) or subsets thereof included in the chunk, whereas the chunk header generally includes metadata relating to the chunk, some or all of which may be derived from the payload. For example, during a secondary copy operation, media agent **144**, storage manager **140**, or other component may divide files into chunks and generate headers for each chunk by processing the files. Headers can include a variety of information such as file and/or volume identifier(s), offset(s), and/or other information associated with the payload data items, a chunk sequence number, etc. Importantly, in addition to being stored with secondary copy **116** on secondary storage device **108**, chunk headers can also be stored to index **153** of the associated media agent(s) **144** and/or to index **150** associated with storage manager **140**. This can be useful for providing faster processing of secondary copies **116** during browsing, restores, or other operations. In some cases, once a chunk is successfully transferred to a secondary storage device **108**, the secondary storage device **108** returns an indication of receipt, e.g., to media agent **144** and/or storage manager **140**, which may update their respective indexes **153**, **150** accordingly. During restore, chunks may be processed (e.g., by media agent **144**) according to the information in the chunk header to reassemble the files.

Data can also be communicated within system **100** in data channels that connect client computing devices **102** to secondary storage devices **108**. These data channels can be referred to as “data streams,” and multiple data streams can be employed to parallelize an information management operation, improving data transfer rate, among other advan-

44

tages. Example data formatting techniques including techniques involving data streaming, chunking, and the use of other data structures in creating secondary copies are described in U.S. Pat. Nos. 7,315,923, 8,156,086, and 8,578,120.

FIGS. 1F and 1G are diagrams of example data streams **170** and **171**, respectively, which may be employed for performing information management operations. Referring to FIG. 1F, data agent **142** forms data stream **170** from source data associated with a client computing device **102** (e.g., primary data **112**). Data stream **170** is composed of multiple pairs of stream header **172** and stream data (or stream payload) **174**. Data streams **170** and **171** shown in the illustrated example are for a single-instanced storage operation, and a stream payload **174** therefore may include both single-instance (SI) data and/or non-SI data. A stream header **172** includes metadata about the stream payload **174**. This metadata may include, for example, a length of the stream payload **174**, an indication of whether the stream payload **174** is encrypted, an indication of whether the stream payload **174** is compressed, an archive file identifier (ID), an indication of whether the stream payload **174** is single instanceable, and an indication of whether the stream payload **174** is a start of a block of data.

Referring to FIG. 1G, data stream **171** has the stream header **172** and stream payload **174** aligned into multiple data blocks. In this example, the data blocks are of size 64 KB. The first two stream header **172** and stream payload **174** pairs comprise a first data block of size 64 KB. The first stream header **172** indicates that the length of the succeeding stream payload **174** is 63 KB and that it is the start of a data block. The next stream header **172** indicates that the succeeding stream payload **174** has a length of 1 KB and that it is not the start of a new data block. Immediately following stream payload **174** is a pair comprising an identifier header **176** and identifier data **178**. The identifier header **176** includes an indication that the succeeding identifier data **178** includes the identifier for the immediately previous data block. The identifier data **178** includes the identifier that the data agent **142** generated for the data block. The data stream **171** also includes other stream header **172** and stream payload **174** pairs, which may be for SI data and/or non-SI data.

FIG. 1H is a diagram illustrating data structures **180** that may be used to store blocks of SI data and non-SI data on a storage device (e.g., secondary storage device **108**). According to certain embodiments, data structures **180** do not form part of a native file system of the storage device. Data structures **180** include one or more volume folders **182**, one or more chunk folders **184/185** within the volume folder **182**, and multiple files within chunk folder **184**. Each chunk folder **184/185** includes a metadata file **186/187**, a metadata index file **188/189**, one or more container files **190/191/193**, and a container index file **192/194**. Metadata file **186/187** stores non-SI data blocks as well as links to SI data blocks stored in container files. Metadata index file **188/189** stores an index to the data in the metadata file **186/187**. Container files **190/191/193** store SI data blocks. Container index file **192/194** stores an index to container files **190/191/193**. Among other things, container index file **192/194** stores an indication of whether a corresponding block in a container file **190/191/193** is referred to by a link in a metadata file **186/187**. For example, data block **B2** in the container file **190** is referred to by a link in metadata file **187** in chunk folder **185**. Accordingly, the corresponding index entry in container index file **192** indicates that data block **B2** in container file **190** is referred to. As another example, data

US 10,210,048 B2

45

block B1 in container file 191 is referred to by a link in metadata file 187, and so the corresponding index entry in container index file 192 indicates that this data block is referred to.

As an example, data structures 180 illustrated in FIG. 1H may have been created as a result of separate secondary copy operations involving two client computing devices 102. For example, a first secondary copy operation on a first client computing device 102 could result in the creation of the first chunk folder 184, and a second secondary copy operation on a second client computing device 102 could result in the creation of the second chunk folder 185. Container files 190/191 in the first chunk folder 184 would contain the blocks of SI data of the first client computing device 102. If the two client computing devices 102 have substantially similar data, the second secondary copy operation on the data of the second client computing device 102 would result in media agent 144 storing primarily links to the data blocks of the first client computing device 102 that are already stored in the container files 190/191. Accordingly, while a first secondary copy operation may result in storing nearly all of the data subject to the operation, subsequent secondary storage operations involving similar data may result in substantial data storage space savings, because links to already stored data blocks can be stored instead of additional instances of data blocks.

If the operating system of the secondary storage computing device 106 on which media agent 144 operates supports sparse files, then when media agent 144 creates container files 190/191/193, it can create them as sparse files. A sparse file is a type of file that may include empty space (e.g., a sparse file may have real data within it, such as at the beginning of the file and/or at the end of the file, but may also have empty space in it that is not storing actual data, such as a contiguous range of bytes all having a value of zero). Having container files 190/191/193 be sparse files allows media agent 144 to free up space in container files 190/191/193 when blocks of data in container files 190/191/193 no longer need to be stored on the storage devices. In some examples, media agent 144 creates a new container file 190/191/193 when a container file 190/191/193 either includes 100 blocks of data or when the size of the container file 190 exceeds 50 MB. In other examples, media agent 144 creates a new container file 190/191/193 when a container file 190/191/193 satisfies other criteria (e.g., it contains from approx. 100 to approx. 1000 blocks or when its size exceeds approximately 50 MB to 1 GB). In some cases, a file on which a secondary copy operation is performed may comprise a large number of data blocks. For example, a 100 MB file may comprise 400 data blocks of size 256 KB. If such a file is to be stored, its data blocks may span more than one container file, or even more than one chunk folder. As another example, a database file of 20 GB may comprise over 40,000 data blocks of size 512 KB. If such a database file is to be stored, its data blocks will likely span multiple container files, multiple chunk folders, and potentially multiple volume folders. Restoring such files may require accessing multiple container files, chunk folders, and/or volume folders to obtain the requisite data blocks. Using Backup Data for Replication and Disaster Recovery (“Live Synchronization”)

There is an increased demand to off-load resource intensive information management tasks (e.g., data replication tasks) away from production devices (e.g., physical or virtual client computing devices) in order to maximize production efficiency. At the same time, enterprises expect

46

access to readily-available up-to-date recovery copies in the event of failure, with little or no production downtime.

FIG. 2A illustrates a system 200 configured to address these and other issues by using backup or other secondary copy data to synchronize a source subsystem 201 (e.g., a production site) with a destination subsystem 203 (e.g., a failover site). Such a technique can be referred to as “live synchronization” and/or “live synchronization replication.” In the illustrated embodiment, the source client computing devices 202a include one or more virtual machines (or “VMs”) executing on one or more corresponding VM host computers 205a, though the source need not be virtualized. The destination site 203 may be at a location that is remote from the production site 201, or may be located in the same data center, without limitation. One or more of the production site 201 and destination site 203 may reside at data centers at known geographic locations, or alternatively may operate “in the cloud.”

The synchronization can be achieved by generally applying an ongoing stream of incremental backups from the source subsystem 201 to the destination subsystem 203, such as according to what can be referred to as an “incremental forever” approach. FIG. 2A illustrates an embodiment of a data flow which may be orchestrated at the direction of one or more storage managers (not shown). At step 1, the source data agent(s) 242a and source media agent(s) 244a work together to write backup or other secondary copies of the primary data generated by the source client computing devices 202a into the source secondary storage device(s) 208a. At step 2, the backup/secondary copies are retrieved by the source media agent(s) 244a from secondary storage. At step 3, source media agent(s) 244a communicate the backup/secondary copies across a network to the destination media agent(s) 244b in destination subsystem 203.

As shown, the data can be copied from source to destination in an incremental fashion, such that only changed blocks are transmitted, and in some cases multiple incremental backups are consolidated at the source so that only the most current changed blocks are transmitted to and applied at the destination. An example of live synchronization of virtual machines using the “incremental forever” approach is found in U.S. Patent Application No. 62/265,339 entitled “Live Synchronization and Management of Virtual Machines across Computing and Virtualization Platforms and Using Live Synchronization to Support Disaster Recovery.” Moreover, a deduplicated copy can be employed to further reduce network traffic from source to destination. For instance, the system can utilize the deduplicated copy techniques described in U.S. Pat. No. 9,239,687, entitled “Systems and Methods for Retaining and Using Data Block Signatures in Data Protection Operations.”

At step 4, destination media agent(s) 244b write the received backup/secondary copy data to the destination secondary storage device(s) 208b. At step 5, the synchronization is completed when the destination media agent(s) and destination data agent(s) 242b restore the backup/secondary copy data to the destination client computing device(s) 202b. The destination client computing device(s) 202b may be kept “warm” awaiting activation in case failure is detected at the source. This synchronization/replication process can incorporate the techniques described in U.S. patent application Ser. No. 14/721,971, entitled “Replication Using Deduplicated Secondary Copy Data.”

Where the incremental backups are applied on a frequent, on-going basis, the synchronized copies can be viewed as mirror or replication copies. Moreover, by applying the

US 10,210,048 B2

47

incremental backups to the destination site **203** using backup or other secondary copy data, the production site **201** is not burdened with the synchronization operations. Because the destination site **203** can be maintained in a synchronized “warm” state, the downtime for switching over from the production site **201** to the destination site **203** is substantially less than with a typical restore from secondary storage. Thus, the production site **201** may flexibly and efficiently fail over, with minimal downtime and with relatively up-to-date data, to a destination site **203**, such as a cloud-based failover site. The destination site **203** can later be reverse synchronized back to the production site **201**, such as after repairs have been implemented or after the failure has passed.

Integrating with the Cloud Using File System Protocols

Given the ubiquity of cloud computing, it can be increasingly useful to provide data protection and other information management services in a scalable, transparent, and highly plug-able fashion. FIG. 2B illustrates an information management system **200** having an architecture that provides such advantages, and incorporates use of a standard file system protocol between primary and secondary storage subsystems **217**, **218**. As shown, the use of the network file system (NFS) protocol (or any another appropriate file system protocol such as that of the Common Internet File System (CIFS)) allows data agent **242** to be moved from the primary storage subsystem **217** to the secondary storage subsystem **218**. For instance, as indicated by the dashed box **206** around data agent **242** and media agent **244**, data agent **242** can co-reside with media agent **244** on the same server (e.g., a secondary storage computing device such as component **106**), or in some other location in secondary storage subsystem **218**.

Where NFS is used, for example, secondary storage subsystem **218** allocates an NFS network path to the client computing device **202** or to one or more target applications **210** running on client computing device **202**. During a backup or other secondary copy operation, the client computing device **202** mounts the designated NFS path and writes data to that NFS path. The NFS path may be obtained from NFS path data **215** stored locally at the client computing device **202**, and which may be a copy of or otherwise derived from NFS path data **219** stored in the secondary storage subsystem **218**.

Write requests issued by client computing device(s) **202** are received by data agent **242** in secondary storage subsystem **218**, which translates the requests and works in conjunction with media agent **244** to process and write data to a secondary storage device(s) **208**, thereby creating a backup or other secondary copy. Storage manager **240** can include a pseudo-client manager **217**, which coordinates the process by, among other things, communicating information relating to client computing device **202** and application **210** (e.g., application type, client computing device identifier, etc.) to data agent **242**, obtaining appropriate NFS path data from the data agent **242** (e.g., NFS path information), and delivering such data to client computing device **202**.

Conversely, during a restore or recovery operation client computing device **202** reads from the designated NFS network path, and the read request is translated by data agent **242**. The data agent **242** then works with media agent **244** to retrieve, re-process (e.g., re-hydrate, decompress, decrypt), and forward the requested data to client computing device **202** using NFS.

By moving specialized software associated with system **200** such as data agent **242** off the client computing devices **202**, the illustrative architecture effectively decouples the

48

client computing devices **202** from the installed components of system **200**, improving both scalability and plug-ability of system **200**. Indeed, the secondary storage subsystem **218** in such environments can be treated simply as a read/write NFS target for primary storage subsystem **217**, without the need for information management software to be installed on client computing devices **202**. As one example, an enterprise implementing a cloud production computing environment can add VM client computing devices **202** without installing and configuring specialized information management software on these VMs. Rather, backups and restores are achieved transparently, where the new VMs simply write to and read from the designated NFS path. An example of integrating with the cloud using file system protocols or so-called “infinite backup” using NFS share is found in U.S. Patent Application No. 62/294,920, entitled “Data Protection Operations Based on Network Path Information.” Examples of improved data restoration scenarios based on network-path information, including using stored backups effectively as primary data sources, may be found in U.S. Patent Application No. 62/297,057, entitled “Data Restoration Operations Based on Network Path Information.”

Highly Scalable Managed Data Pool Architecture

Enterprises are seeing explosive data growth in recent years, often from various applications running in geographically distributed locations. FIG. 2C shows a block diagram of an example of a highly scalable, managed data pool architecture useful in accommodating such data growth. The illustrated system **200**, which may be referred to as a “web-scale” architecture according to certain embodiments, can be readily incorporated into both open compute/storage and common-cloud architectures.

The illustrated system **200** includes a grid **245** of media agents **244** logically organized into a control tier **231** and a secondary or storage tier **233**. Media agents assigned to the storage tier **233** can be configured to manage a secondary storage pool **208** as a deduplication store, and be configured to receive client write and read requests from the primary storage subsystem **217**, and direct those requests to the secondary tier **233** for servicing. For instance, media agents CMA1-CMA3 in the control tier **231** maintain and consult one or more deduplication databases **247**, which can include deduplication information (e.g., data block hashes, data block links, file containers for deduplicated files, etc.) sufficient to read deduplicated files from secondary storage pool **208** and write deduplicated files to secondary storage pool **208**. For instance, system **200** can incorporate any of the deduplication systems and methods shown and described in U.S. Pat. No. 9,020,900, entitled “Distributed Deduplicated Storage System,” and U.S. Pat. Pub. No. 2014/0201170, entitled “High Availability Distributed Deduplicated Storage System.”

Media agents SMA1-SMA6 assigned to the secondary tier **233** receive write and read requests from media agents CMA1-CMA3 in control tier **231**, and access secondary storage pool **208** to service those requests. Media agents CMA1-CMA3 in control tier **231** can also communicate with secondary storage pool **208**, and may execute read and write requests themselves (e.g., in response to requests from other control media agents CMA1-CMA3) in addition to issuing requests to media agents in secondary tier **233**. Moreover, while shown as separate from the secondary storage pool **208**, deduplication database(s) **247** can in some cases reside in storage devices in secondary storage pool **208**.

As shown, each of the media agents **244** (e.g., CMA1-CMA3, SMA1-SMA6, etc.) in grid **245** can be allocated a corresponding dedicated partition **251A-251I**, respectively, in secondary storage pool **208**. Each partition **251** can include a first portion **253** containing data associated with (e.g., stored by) media agent **244** corresponding to the respective partition **251**. System **200** can also implement a desired level of replication, thereby providing redundancy in the event of a failure of a media agent **244** in grid **245**. Along these lines, each partition **251** can further include a second portion **255** storing one or more replication copies of the data associated with one or more other media agents **244** in the grid.

System **200** can also be configured to allow for seamless addition of media agents **244** to grid **245** via automatic configuration. As one illustrative example, a storage manager (not shown) or other appropriate component may determine that it is appropriate to add an additional node to control tier **231**, and perform some or all of the following: (i) assess the capabilities of a newly added or otherwise available computing device as satisfying a minimum criteria to be configured as or hosting a media agent in control tier **231**; (ii) confirm that a sufficient amount of the appropriate type of storage exists to support an additional node in control tier **231** (e.g., enough disk drive capacity exists in storage pool **208** to support an additional deduplication database **247**); (iii) install appropriate media agent software on the computing device and configure the computing device according to a pre-determined template; (iv) establish a partition **251** in the storage pool **208** dedicated to the newly established media agent **244**; and (v) build any appropriate data structures (e.g., an instance of deduplication database **247**). An example of highly scalable managed data pool architecture or so-called web-scale architecture for storage and data management is found in U.S. Patent Application No. 62/273,286 entitled "Redundant and Robust Distributed Deduplication Data Storage System."

The embodiments and components thereof disclosed in FIGS. 2A, 2B, and 2C, as well as those in FIGS. 1A-1H, may be implemented in any combination and permutation to satisfy data storage management and information management needs at one or more locations and/or data centers. Selective Snapshot and Backup Copy Operations for Individual Virtual Machines in a Shared Storage

As discussed herein, virtual machines are in common use. In general terms, a virtual machine is computer software operating on a host computer hardware system so as to emulate a guest computer hardware system. For example, a virtual machine can emulate, in software, a guest computer's processor, memory, hard disk drive, network connection, and/or other hardware resources. Each virtual machine can run its own operating system and applications such that the user of a virtual machine can have substantially the same experience as he or she would have on dedicated hardware. Multiple virtual machines can operate on the same host computer hardware system (e.g., a virtual machine server), thus allowing common hardware resources to be efficiently and flexibly shared amongst users of the virtual guest machines.

A hypervisor is the name for software and/or hardware which creates and runs virtual machines on a host computer hardware system. As discussed herein, an example hypervisor is ESX Server, available from VMware, Inc. of Palo Alto, Calif. A hypervisor typically provisions resources to each virtual machine, such as a virtual processor, an operating system, virtual memory, a virtual network device, and a virtual storage device, such as a virtual disk. A virtual

machine reads data from, and writes data to, its virtual disk similarly to how a physical machine reads data from and writes data to a physical disk.

Virtual storage disks are files stored in the file system of the physical host system. The files are called virtual machine disk files ("VMDK" in VMware terminology). Typically, a hypervisor stores the virtual machine disk files corresponding to many different and independent virtual machines in a common datastore. For example, VMware's ESX Server provides the Virtual Machine File System (VMFS) as a common datastore for the storage of virtual machine disk files.

FIG. 3 illustrates an example virtual machine system **300**. The system **300** includes a hypervisor **302** which creates and runs multiple virtual machines VM_1, VM_2, \dots, VM_N (respectively **310a**, **310b**, **310c**). The hypervisor **302** is communicatively connected to a primary storage system **304** via a data path **372**. The primary storage system **304** can include one or more physical computer storage devices. In some embodiments, the primary storage system **304** is a storage area network (SAN). The hypervisor **302** can be communicatively connected to the primary storage system **304** via, for example, a Small Computer System Interface (SCSI) over a Fibre Channel network or an Internet Computer System Interface (iSCSI) over the Internet. Other connections and protocols can also be used to communicatively connect the hypervisor **302** and the primary storage system **304**. The hypervisor **302** is an example of a client computing device **102** discussed herein with respect to FIGS. 1-2 and can include any of the features and perform any of the functions of that device.

Typically, the primary storage system **304** includes one or more logical units, each identified by a logical unit number (LUN). Each logical unit is a separate storage volume accessible by an independent file system. Within the logical unit, the hypervisor **302** provides a shared virtual machine datastore. Typically there is a one-to-one correspondence between a shared datastore (e.g., a Virtual Machine File System) and a logical unit. The shared virtual machine datastore is used to store the data corresponding to multiple independent virtual machines created and operated by the hypervisor **302**. Again, the data for each virtual machine can be stored in a virtual machine disk file. As shown in FIG. 3, the shared virtual machine datastore includes a first virtual disk **312a** which stores the data corresponding to virtual machine VM_1 . It also includes virtual disks **312b** and **312c** which respectively store the data corresponding to virtual machines VM_2 and VM_N . Although FIG. 3 illustrates three virtual machines, it is intended to represent that any arbitrary number of independent virtual machines can be operated by the hypervisor **302** and stored in the shared datastore of the LUN.

Just as it is a useful practice to create snapshot and backup copies of the data stored on the disk of a physical computer hardware system, it is also useful to create snapshot and/or backup copies of virtual machines. This can be done by making a snapshot and/or backup copy of a virtual machine's virtual disk file. But in conventional virtual machine systems such as the one illustrated in FIG. 3 there may be difficulties with selectively making snapshot and/or backup copies of individual virtual machine disks in the shared datastore. As already discussed, typically a shared virtual machine datastore corresponds to a logical unit of storage. And, as its name suggests, a logical unit has conventionally been the smallest unit of addressable storage in such systems. As a result, snapshots are typically performed at the logical unit level. In other words, storage

US 10,210,048 B2

51

systems used in virtual machine systems may only allow or be capable of performing a snapshot of an entire logical unit, as opposed to selectively snapshotting some sub-portion of the logical unit. The result is that it may not be possible to individually snapshot a selected virtual machine; instead, it may only be possible to create a snapshot of the entire shared virtual machine datastore in the logical unit. This naturally results in a snapshot of all the virtual machine disks in the shared datastore. Thus, it may be necessary to snapshot large numbers of virtual machines together or not at all. This is an undesirable restriction which results in inefficient snapshot and backup copy operations as they pertain to virtual machine systems. FIG. 4 illustrates an improved virtual machine system which can overcome these difficulties.

FIG. 4 illustrates an example virtual machine system 400 with improved capabilities to selectively and individually make snapshot and/or backup copies of virtual machines in a shared storage. The system 400 accomplishes this in part by creating separate volumes for the virtual disks of the various virtual machines. In the case of VMware, for example, a separate volume can be created for each virtual machine disk file (VMDK). The virtual machine volumes are independent of the type of underlying physical storage. They do not necessarily need to be formatted with a file system and can be configured on demand. As a result, pre-allocated LUNs and volumes are not required. In the system 400, the virtual machine volume is the primary unit of data management. Thus, snapshot and/or backup copy and other storage operations can be performed with virtual machine-level granularity rather than LUN-level granularity. For instance, the system 400 permits snapshotting of individual volumes corresponding to each virtual machine. The components of the virtual machine system 400 will now be discussed.

The system 400 includes a hypervisor 402 which creates and runs multiple virtual machines VM_1, VM_2, \dots, VM_N (respectively 410a, 410b, 410c). The hypervisor 402 is communicatively connected to a primary storage system 404 via a data path 472. The primary storage system 404 can include one or more physical computer storage devices. In some embodiments, the primary storage system 404 is a storage area network (SAN). The hypervisor 402 can be communicatively connected to the primary storage system 404 via, for example, SCSI over a Fibre Channel network or iSCSI over the Internet. Other connections and protocols can also be used to communicatively connect the hypervisor 402 and the primary storage system 404. The hypervisor 402 is an example of a client computing device 102 discussed herein with respect to FIGS. 1-2 and can include any of the features and perform any of the functions of that device.

As before, the primary storage system 404 is used to store the data corresponding to multiple independent virtual machines created and operated by the hypervisor 402. The data stored in the primary storage system 404 is an example of primary data (as discussed herein). For example, the data in the primary storage system 404 is "live" data generated by the operating system and/or applications of the corresponding virtual machine, generally in the native format of the source application. The data for each virtual machine can be stored in a virtual machine disk file. As shown in FIG. 4, the shared primary storage system 404 includes a first virtual disk 412a which stores the data corresponding to virtual machine VM_1 . It also includes virtual disks 412b and 412c which respectively store the data corresponding to virtual machines VM_2 and VM_N . Although FIG. 4 illustrates three virtual machines, the system 400 can operate and store any arbitrary number of virtual machines.

52

Unlike the system 300 shown in FIG. 3, the system 400 shown in FIG. 4 stores the data corresponding to each virtual machine in separate volumes. The provisioning of separate volumes for each virtual machine is made possible in part by a bi-directional path 470 for control information to be exchanged between the hypervisor 402 and the primary storage system 404. An example of such a control path 470 for use with VMware's ESX hypervisor is the vSphere APIs for Storage Awareness (VASA) 2.0. This control path is a conduit for configuration information.

To understand the benefit of this control path 470, it is helpful to understand how storage is conventionally provisioned for virtual machines in a system such as the one illustrated in FIG. 3. Conventionally, storage administrator personnel would pre-allocate LUNs with various configuration parameters (e.g., performance, availability, RAID level, snapshot capability, compression, replication capability, etc.) based on the potential needs of different virtual machines provided by the hypervisor 302. Upon creation of a virtual machine, the hypervisor 302 would then assign it to an available pre-allocated LUN whose capabilities matched the requirements of the virtual machine.

But the bi-directional control path 470 illustrated in FIG. 4 contributes to allowing the system 400 to operate differently. Instead of assigning virtual machines to existing LUNs, the bi-directional control path 470 allows the primary storage system 404 to broadcast its capabilities and features (e.g., performance, availability, RAID level, snapshot capability, compression, replication capability, etc.) to the hypervisor 402. Then, when the hypervisor 402 creates a virtual machine, it can send the storage requirements of that particular virtual machine to the primary storage system 404. In other words, the bi-directional control path allows storage capabilities to be surfaced up from the primary storage system 404 to the hypervisor 402 and storage requirements of virtual machines to be pushed down from the hypervisor to the primary storage system 404. The primary storage system 404 is designed with the capability to create a separate volume for each virtual machine—in accordance with the storage requirements specified by the hypervisor—on demand.

In addition to the bi-directional control path 470 between the hypervisor 402 and the primary storage system 404, the data path 472 is used for the exchange of the application data to be stored in the primary storage system 404. A demultiplexor 460 can be provided in this data path 472 to assist with storage I/O between the hypervisor 402 and the primary storage system 404. This can improve scalability if, for example, the hypervisor 402 has a limit on the number of storage units it can work with (e.g., ESX server is limited to 256 LUNs). Such limits are not typically problematic when many virtual machine disk files are stored in a single logical unit, but they can be limiting when there is one volume per virtual machine. Given the potentially-large number of separate volumes that can be created in the primary storage system 404, the demultiplexor 460 can act as a single connection to the primary storage system 404 for I/O operations to eliminate scalability problems.

The virtual machine system 400 also includes a secondary storage system 408. The secondary storage system 408 is used to store backup copies (416a, 416b, 416c) of the virtual machines VM_1, VM_2, \dots, VM_N . The secondary storage system 408 is made up of hardware computer storage devices which are separate from those which make up the primary storage system 404. Backup copies may be stored on media with slower retrieval times than primary storage system.

The virtual machine system **400** also includes a virtual server agent **440**. The virtual server agent **440** generally initiates, performs, coordinates, and/or controls snapshot and backup copying operations and other information management operations performed by the system **400**. For example, the virtual server agent **440** can control the selective snapshotting of individual virtual machines as well as selectively making backup copies of individual virtual machines in the secondary storage system **408**. These snapshot and backup copying operations can be performed according to an example method illustrated in FIG. 5. The virtual server agent **440** may be a software module or other application hosted by a suitable computing device with a processor and memory configured to execute instructions for implementing the features and performing the tasks described herein. In some embodiments, the virtual server agent **440** is external to the hypervisor **402** and is implemented with a separate hardware device. In other embodiments, the virtual server agent **440** can itself be a virtual machine operating on the hypervisor **402**.

As shown by the arrowed lines **414** in FIG. 4, the virtual server agent **440** may communicate with, instruct, and/or control other elements of the system **400**. In this manner, the virtual server agent **440** manages the operation of various hardware and software components in the system **400**. In some embodiments, control information originates from the virtual server agent **440** and status, reports, metadata, and other control information are transmitted to the virtual server agent **440** by the managed components, whereas payload data are generally communicated between the managed components themselves, such as the hypervisor **402**, the primary storage system **404**, and the secondary storage system **408**. Control information can generally include parameters and instructions for carrying out snapshot and backup copying and other information management operations (e.g., instructions to perform a task associated with an operation, timing information specifying when to initiate a task, data path information specifying what components to communicate with or access in carrying out an operation, and the like).

The virtual server agent **440** may maintain a database **446** of metadata associated with snapshot and backup copies of the virtual machine data stored in the primary storage system **404**. The database **446** is stored in computer memory accessible by the virtual server agent **440**.

The virtual server agent **440** can also include a user interface, such as a graphical user interface (GUI), an application program interface (API), and/or other interactive interface(s) through which users and system processes can retrieve information about the status of snapshot and backup copying and other information management operations, or issue instructions to the virtual server agent **440** and other components. Via the user interface, users may issue instructions to the components in the system **400** regarding performance of snapshot and backup copy operations.

FIG. 5 illustrates an example method **500** according to which the virtual server agent **440** can control snapshot and backup copying of the virtual machines VM_1, VM_2, \dots, VM_N in the virtual machine system **400** illustrated in FIG. 4. Advantageously, the virtual server agent **440** can be used to selectively perform snapshot and backup copy operations on any specific virtual machine stored in the shared primary storage system **404** without necessitating that such operations be performed for any other virtual machine besides the selected one.

The method **500** begins at block **510** where the virtual server agent **440** issues a command to perform a snapshot

copy operation for a selected virtual machine operated by the hypervisor **402** and stored in the primary storage system **404**. The command can be transmitted from the virtual server agent **440** to the hypervisor **402** and/or the primary storage system **404** (e.g., via the control path **470**). In the case of a system implemented using VMware software, the command can be issued via the vStorage API for Data Protection (VADP).

As discussed herein, a snapshot copy operation can provide a relatively lightweight, efficient mechanism for protecting data. From an end user viewpoint, a snapshot may be thought of as an “instant” image of virtual machine data (**412a**, **412b**, **412c**) at a given point in time. In some embodiments, a snapshot may generally capture the directory structure of the primary data at a particular moment in time and may also preserve file attributes and contents. A snapshot can typically be created relatively quickly (e.g., substantially instantly) using a relatively small amount of storage space. Snapshots typically do not actually create another physical copy of all the data as it existed at the particular point in time, but may simply create pointers that map files and directories to specific memory locations where the data resides as it existed at the particular point in time. An initial snapshot may use only a small amount of disk space needed to record a mapping or other data structure representing or otherwise tracking the blocks that correspond to the current state of the file system. Additional disk space is usually required only when files and directories change later on. Furthermore, when files change, typically only the pointers which map to blocks are copied, not the blocks themselves. By electing to restore primary data (**412a**, **412b**, **412c**) from a snapshot taken at a given point in time, users may return a virtual machine to its state when the snapshot was taken.

In some embodiments, the snapshot copy operation can be a snapshot of a selected individual virtual machine. The specific virtual machine to which the snapshot copy operation is to be applied can be selected by, for example, a user via the user interface of the virtual server agent **440** or by an automated process according to a schedule, profile, etc. (A list of the virtual machines being operated by the system **400** can be provided to the virtual server agent by the hypervisor **402**.) In addition, the command to perform the snapshot copy operation can be initiated by the user or according to the automated process.

At block **520**, once the command is received by the hypervisor and/or the primary storage system **404**, the command is executed. For example, the primary storage system **404** can perform a hardware-level snapshot of the specific volume corresponding to the selected virtual machine. This can be done according to any of the snapshot techniques discussed herein and the resulting snapshot can be stored in the primary storage system **404**. As already discussed, the snapshot operation can be performed for a single virtual machine without requiring that other virtual machine volumes stored in the primary storage system **404** also be snapshotted. Once the snapshot copy operation has been completed, the hypervisor **402** and/or the primary storage system **404** transmits metadata regarding the snapshot copy to the virtual server agent **440**. The metadata can include, for example, an identifier for the snapshot (e.g., an ID number). In some embodiments, the metadata can also, or alternatively, include an identifier for the virtual machine corresponding to the snapshot, date and time stamps, a pointer to the storage location of the snapshot, etc.

At block **530**, the virtual server agent **440** stores the snapshot metadata in its database **446**. The database **446** can

US 10,210,048 B2

55

store snapshot metadata and present this information to a user via the user interface of the virtual server agent **440**, an example of which is illustrated in FIG. **6**.

FIG. **6** depicts an illustrative graphical user interface **600** showing snapshot metadata collected by the virtual server agent in the illustrative system of FIG. **4**. As illustrated, the user interface presents to a user a list of virtual machine snapshots that have been performed. Each snapshot is given an identifier, which is provided to the virtual server agent **440** by the hypervisor **402** in response to completion of a successful snapshot copy command. Examples of snapshot identifiers are shown in the first column of the user interface **600**. In the second column, the user interface **600** shows the virtual machine to which each snapshot corresponds. The third and fourth columns show date and time stamps of when the snapshot copy was created. Finally, the last column of the user interface **600** shows the storage path where the snapshot copy is located. It should be understood, however, that some embodiments may use some of the illustrated types of metadata and not others.

At block **540** of FIG. **5**, the virtual server agent **440** can also be used to control backup copy operations. A backup copy operation can be performed using the stored metadata from one or more snapshot copy operations. In particular, the virtual server agent **440** can receive a selection of one of the virtual machines for which to create a backup copy. The virtual server agent **440** can also receive a selection of a snapshot copy (whether existing or newly-created) to use for creating the backup copy. The backup copy can be made according to any of the techniques discussed herein. A backup copy operation creates a copy of a version of primary data at a particular point in time (e.g., one or more files or other data units). A backup generally involves maintaining a version of the copied primary data (**412a**, **412b**, **412c**) as well as backup copies (**416a**, **416b**, **416c**). Further, a backup copy in some embodiments is generally stored in a form that is different from the native format (e.g., a backup format). This contrasts to the primary data which may instead be stored in a format native to the source application(s). In some embodiments, backup copies can be stored in a format in which the data is compressed, encrypted, deduplicated, and/or otherwise modified from the original native application format. For example, a backup copy may be stored in a compressed backup format that facilitates efficient long-term storage. Backup copies can have relatively long retention periods as compared to primary data, which is generally highly changeable.

Metadata for the completed snapshots can be presented to a user via a user interface **600** such as the one illustrated in FIG. **6**. With this interface **600** a user can select a virtual machine for which he or she would like to create a backup copy. In the case illustrated in FIG. **6**, the user has selected to create a backup copy of virtual machine VM₁. Accordingly, the user interface **600** highlights the available snapshots which have been taken for that virtual machine by placing a box around those records. In this case, two snapshots are available: one from January 1st at 9:00 am and another from January 2nd at 3:00 pm. The virtual server agent then receives input from the user regarding a selection of a specific snapshot to use for completion of the backup copy operation.

At block **550** of FIG. **5**, the virtual server agent issues a command to perform the selected backup copy operation. This command may be transmitted to the hypervisor **402** and/or the primary storage system **404**. The command may be accompanied by the stored metadata for the snapshot copy which has been selected to be used in the creation of

56

a backup copy of the selected virtual machine. For example, the virtual server agent **440** may transmit a snapshot identifier for the selected snapshot to the hypervisor **402**. The hypervisor **402** may use the snapshot identifier to access the selected snapshot in the primary storage system **404**. The selected snapshot can then be used to create a backup copy of the selected virtual machine. The backup copy is stored in the secondary storage system **408** (e.g., a tape drive).

Although certain features and functions of the virtual server agent have just been discussed, it should be understood that the virtual server agent **440** is an example of a storage manager **140**, as discussed herein with respect to FIGS. **1-2**. As such, the virtual server agent **440** can include any of the features and perform any of the functions of the storage manager **140** disclosed herein. In addition, the system **400** can include one or more data agents **142**, media agents **144**, secondary storage computing devices **106**, etc., as discussed herein with respect to FIGS. **1-2**. The virtual server agent can interact with any of these devices in the ways discussed herein.

In regard to the figures described herein, other embodiments are possible within the scope of the present invention, such that the above-recited components, steps, blocks, operations, and/or messages/requests/queries/instructions are differently arranged, sequenced, sub-divided, organized, and/or combined. In some embodiments, a different component may initiate or execute a given operation.

Example Embodiments

Some example enumerated embodiments of the present invention are recited in this section in the form of methods, systems, and non-transitory computer-readable media, without limitation.

In some embodiments, a system comprises: a hypervisor configured to create and operate a plurality of virtual machines; one or more shared physical computer storage devices communicatively coupled to the hypervisor to store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; and a virtual server agent configured to issue a command to the hypervisor to perform a snapshot copy operation for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

In some embodiments, the virtual server agent is external to the hypervisor and is implemented on separate hardware.

In some embodiments, the snapshot copy comprises virtual machine data stored in its native format.

In some embodiments, the virtual server agent is further configured to store metadata corresponding to the snapshot copy operation when the snapshot copy operation is completed.

In some embodiments, the metadata comprises a snapshot identifier.

In some embodiments, the virtual server agent is further configured to issue a command to perform a backup copy operation using the metadata corresponding to the completed snapshot copy operation.

In some embodiments, the backup copy operation causes a backup copy of the virtual machine to be stored on one or more external computer storage devices.

US 10,210,048 B2

57

In some embodiments, the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

In some embodiments, the hypervisor and the one or more shared physical computer storage devices are communicatively coupled by a bi-directional control path which allows storage capabilities to be provided from the one or more shared physical computer storage devices to the hypervisor and storage requirements of the virtual machines to be provided from the hypervisor to the one or more shared physical computer storage devices.

In some embodiments, the one or more shared physical computer storage devices are configured to create a storage volume for a newly-created virtual machine on demand according to storage requirements for the newly-created virtual machine.

In some embodiments, the system comprises a demultiplexor to communicate input/output commands between the hypervisor and the one or more shared physical computer storage devices.

In some embodiments, the virtual server agent comprises one of the plurality of virtual machines operated by the hypervisor.

In some embodiments, the one or more shared physical storage devices comprise a storage area network.

In some embodiments, a method comprises: by a hypervisor, creating and operating a plurality of virtual machines; by one or more shared physical computer storage devices communicatively coupled to the hypervisor, storing the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; and by a virtual server agent, issuing a command to the hypervisor to perform a snapshot copy operation for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

In some embodiments, the snapshot copy comprises virtual machine data stored in its native format.

In some embodiments, the method further comprises, by the virtual server agent, storing metadata corresponding to the snapshot copy operation when the snapshot copy operation is completed.

In some embodiments, the metadata comprises a snapshot identifier.

In some embodiments, the method further comprises, by the virtual server agent, issuing a command to perform a backup copy operation using the metadata corresponding to the completed snapshot copy operation.

In some embodiments, the method further comprises, by the virtual server agent, causing the backup copy of the virtual machine to be stored on one or more external computer storage devices.

In some embodiments, the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

In some embodiments, virtual server agent comprises: a memory for storing instructions to carry out a method comprising issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely correspond-

58

ing to one of the virtual machines; receiving metadata corresponding to the completed snapshot copy operation from the hypervisor; and storing the metadata in a database, wherein the snapshot copy operation is performed for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices; and a processor to execute the instructions.

In some embodiments, the virtual server agent is external to the hypervisor and is implemented on separate hardware.

In some embodiments, the snapshot copy comprises virtual machine data stored in its native format.

In some embodiments, the metadata comprises a snapshot identifier.

In some embodiments, the instructions further cause the processor to issue a command to the hypervisor to perform a backup copy operation using the metadata corresponding to the completed snapshot copy operation.

In some embodiments, the backup copy operation causes a backup copy of the virtual machine to be stored on one or more external computer storage devices.

In some embodiments, the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

In some embodiments, a virtual server agent method comprises: by the virtual server agent, issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; by the virtual server agent, receiving metadata corresponding to the completed snapshot copy operation from the hypervisor; and by the virtual server agent, storing the metadata in a database, wherein the snapshot copy operation is performed for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

In some embodiments, the snapshot copy comprises virtual machine data stored in its native format.

In some embodiments, the metadata comprises a snapshot identifier.

In some embodiments, the method further comprises issuing a command to the hypervisor to perform a backup copy operation using the metadata corresponding to the completed snapshot copy operation.

In some embodiments, the backup copy operation causes a backup copy of the virtual machine to be stored on one or more external computer storage devices.

In some embodiments, the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

In some embodiments, a non-transitory computer readable medium stores instructions which when executed by at least one computing device perform a virtual server agent method comprising: by the virtual server agent, issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the

US 10,210,048 B2

59

one or more shared physical computer storage devices, each storage volume uniquely corresponding to one of the virtual machines; by the virtual server agent, receiving metadata corresponding to the completed snapshot copy operation from the hypervisor; and by the virtual server agent, storing the metadata in a database, wherein the snapshot copy operation is performed for a selected one of the plurality of virtual machines without performing the snapshot copy operation for any other unselected virtual machine in the one or more shared physical computer storage devices.

In some embodiments, the snapshot copy comprises virtual machine data stored in its native format.

In some embodiments, the metadata comprises a snapshot identifier.

In some embodiments, the instructions further cause the computing device to issue a command to the hypervisor to perform a backup copy operation using the metadata corresponding to the completed snapshot copy operation.

In some embodiments, the backup copy operation causes a backup copy of the virtual machine to be stored on one or more external computer storage devices.

In some embodiments, the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

In some embodiments, the instructions cause the method to be performed by a virtual machine operated by the hypervisor.

In some embodiments, a virtual server agent comprises: a memory for storing instructions to carry out a method comprising commanding an external system to create a snapshot copy of any one of a plurality of virtual machines stored in the external system; receiving metadata regarding the completed snapshot copy from the external system; storing the metadata in a database; receiving a selection of any of one or more completed snapshot copies to use for creating a backup copy of any of the plurality of virtual machines; and commanding the external system to create a backup copy of a selected virtual machine using metadata corresponding to the selected snapshot; and a processor to execute the instructions.

In some embodiments, the snapshot copy comprises virtual machine data stored in its native format.

In some embodiments, the metadata comprises a snapshot identifier.

In some embodiments, the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

In some embodiments, the backup copy of the selected virtual machine is stored in an external storage device.

In some embodiments, the command to create the snapshot copy causes the external system to create the snapshot copy of a selected one of the plurality of virtual machines without creating a snapshot copy of any other unselected virtual machine.

In some embodiments, the command to create the backup copy causes the external system to create the backup copy of the selected one of the plurality of virtual machines without creating a backup copy of any other unselected virtual machine.

In some embodiments, a method performed by a virtual server agent comprises: by the virtual server agent, commanding an external system to create a snapshot copy of any one of a plurality of virtual machines stored in the external system; by the virtual server agent, receiving metadata regarding the completed snapshot copy from the external system; by the virtual server agent, storing the metadata in a database; by the virtual server agent, receiving a selection

60

of any of one or more completed snapshot copies to use for creating a backup copy of any of the plurality of virtual machines; and by the virtual server agent, commanding the external system to create a backup copy of a selected virtual machine using metadata corresponding to the selected snapshot.

In some embodiments, the snapshot copy comprises virtual machine data stored in its native format.

In some embodiments, the metadata comprises a snapshot identifier.

In some embodiments, the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

In some embodiments, the backup copy of the selected virtual machine is stored in an external storage device.

In some embodiments, commanding the external system to create the snapshot copy causes the external system to create the snapshot copy of a selected one of the plurality of virtual machines without creating a snapshot copy of any other unselected virtual machine.

In some embodiments, commanding the external system to create the backup copy causes the external system to create the backup copy of the selected one of the plurality of virtual machines without creating a backup copy of any other unselected virtual machine.

In some embodiments, a non-transitory computer readable medium stores instructions which when executed by at least one computing device perform a virtual server agent method comprising: by the virtual server agent, commanding an external system to create a snapshot copy of any one of a plurality of virtual machines stored in the external system; by the virtual server agent, receiving metadata regarding the completed snapshot copy from the external system; by the virtual server agent, storing the metadata in a database; by the virtual server agent, receiving a selection of any of one or more completed snapshot copies to use for creating a backup copy of any of the plurality of virtual machines; and by the virtual server agent, commanding the external system to create a backup copy of a selected virtual machine using metadata corresponding to the selected snapshot.

In some embodiments, the snapshot copy comprises virtual machine data stored in its native format.

In some embodiments, the metadata comprises a snapshot identifier.

In some embodiments, the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

In some embodiments, commanding the external system to create the snapshot copy causes the external system to create the snapshot copy of a selected one of the plurality of virtual machines without creating a snapshot copy of any other unselected virtual machine.

In some embodiments, commanding the external system to create the backup copy causes the external system to create the backup copy of the selected one of the plurality of virtual machines without creating a backup copy of any other unselected virtual machine.

In other embodiments, a system or systems may operate according to one or more of the methods and/or computer-readable media recited in the preceding paragraphs. In yet other embodiments, a method or methods may operate according to one or more of the systems and/or computer-readable media recited in the preceding paragraphs. In yet more embodiments, a computer-readable medium or media, excluding transitory propagating signals, may cause one or more computing devices having one or more processors and

US 10,210,048 B2

61

non-transitory computer-readable memory to operate according to one or more of the systems and/or methods recited in the preceding paragraphs.

Terminology

Conditional language, such as, among others, “can,” “could,” “might,” or “may,” unless specifically stated otherwise, or otherwise understood within the context as used, is generally intended to convey that certain embodiments include, while other embodiments do not include, certain features, elements and/or steps. Thus, such conditional language is not generally intended to imply that features, elements and/or steps are in any way required for one or more embodiments or that one or more embodiments necessarily include logic for deciding, with or without user input or prompting, whether these features, elements and/or steps are included or are to be performed in any particular embodiment.

Unless the context clearly requires otherwise, throughout the description and the claims, the words “comprise,” “comprising,” and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense, i.e., in the sense of “including, but not limited to.” As used herein, the terms “connected,” “coupled,” or any variant thereof means any connection or coupling, either direct or indirect, between two or more elements; the coupling or connection between the elements can be physical, logical, or a combination thereof. Additionally, the words “herein,” “above,” “below,” and words of similar import, when used in this application, refer to this application as a whole and not to any particular portions of this application. Where the context permits, words using the singular or plural number may also include the plural or singular number respectively. The word “or” in reference to a list of two or more items, covers all of the following interpretations of the word: any one of the items in the list, all of the items in the list, and any combination of the items in the list. Likewise the term “and/or” in reference to a list of two or more items, covers all of the following interpretations of the word: any one of the items in the list, all of the items in the list, and any combination of the items in the list.

In some embodiments, certain operations, acts, events, or functions of any of the algorithms described herein can be performed in a different sequence, can be added, merged, or left out altogether (e.g., not all are necessary for the practice of the algorithms). In certain embodiments, operations, acts, functions, or events can be performed concurrently, e.g., through multi-threaded processing, interrupt processing, or multiple processors or processor cores or on other parallel architectures, rather than sequentially.

Systems and modules described herein may comprise software, firmware, hardware, or any combination(s) of software, firmware, or hardware suitable for the purposes described. Software and other modules may reside and execute on servers, workstations, personal computers, computerized tablets, PDAs, and other computing devices suitable for the purposes described herein. Software and other modules may be accessible via local computer memory, via a network, via a browser, or via other means suitable for the purposes described herein. Data structures described herein may comprise computer files, variables, programming arrays, programming structures, or any electronic information storage schemes or methods, or any combinations thereof, suitable for the purposes described herein. User interface elements described herein may comprise elements

62

from graphical user interfaces, interactive voice response, command line interfaces, and other suitable interfaces.

Further, processing of the various components of the illustrated systems can be distributed across multiple machines, networks, and other computing resources. Two or more components of a system can be combined into fewer components. Various components of the illustrated systems can be implemented in one or more virtual machines, rather than in dedicated computer hardware systems and/or computing devices. Likewise, the data repositories shown can represent physical and/or logical data storage, including, e.g., storage area networks or other distributed storage systems. Moreover, in some embodiments the connections between the components shown represent possible paths of data flow, rather than actual connections between hardware. While some examples of possible connections are shown, any of the subset of the components shown can communicate with any other subset of components in various implementations.

Embodiments are also described above with reference to flow chart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products. Each block of the flow chart illustrations and/or block diagrams, and combinations of blocks in the flow chart illustrations and/or block diagrams, may be implemented by computer program instructions. Such instructions may be provided to a processor of a general purpose computer, special purpose computer, specially-equipped computer (e.g., comprising a high-performance database server, a graphics subsystem, etc.) or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor(s) of the computer or other programmable data processing apparatus, create means for implementing the acts specified in the flow chart and/or block diagram block or blocks. These computer program instructions may also be stored in a non-transitory computer-readable memory that can direct a computer or other programmable data processing apparatus to operate in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the acts specified in the flow chart and/or block diagram block or blocks. The computer program instructions may also be loaded to a computing device or other programmable data processing apparatus to cause operations to be performed on the computing device or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computing device or other programmable apparatus provide steps for implementing the acts specified in the flow chart and/or block diagram block or blocks.

Any patents and applications and other references noted above, including any that may be listed in accompanying filing papers, are incorporated herein by reference. Aspects of the invention can be modified, if necessary, to employ the systems, functions, and concepts of the various references described above to provide yet further implementations of the invention. These and other changes can be made to the invention in light of the above Detailed Description. While the above description describes certain examples of the invention, and describes the best mode contemplated, no matter how detailed the above appears in text, the invention can be practiced in many ways. Details of the system may vary considerably in its specific implementation, while still being encompassed by the invention disclosed herein. As noted above, particular terminology used when describing certain features or aspects of the invention should not be

US 10,210,048 B2

63

taken to imply that the terminology is being redefined herein to be restricted to any specific characteristics, features, or aspects of the invention with which that terminology is associated. In general, the terms used in the following claims should not be construed to limit the invention to the specific examples disclosed in the specification, unless the above Detailed Description section explicitly defines such terms. Accordingly, the actual scope of the invention encompasses not only the disclosed examples, but also all equivalent ways of practicing or implementing the invention under the claims.

To reduce the number of claims, certain aspects of the invention are presented below in certain claim forms, but the applicant contemplates other aspects of the invention in any number of claim forms. For example, while only one aspect of the invention is recited as a means-plus-function claim under 35 U.S.C. sec. 112(f) (AIA), other aspects may likewise be embodied as a means-plus-function claim, or in other forms, such as being embodied in a computer-readable medium. Any claims intended to be treated under 35 U.S.C. § 112(f) will begin with the words “means for,” but use of the term “for” in any other context is not intended to invoke treatment under 35 U.S.C. § 112(f). Accordingly, the applicant reserves the right to pursue additional claims after filing this application, in either this application or in a continuing application.

What is claimed is:

1. A virtual server agent comprising:
a memory for storing instructions to carry out a method comprising:
issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines,
wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, and
wherein each storage volume uniquely corresponds to one of the virtual machines in the plurality of virtual machines,
receiving metadata corresponding to the completed snapshot copy operation from the hypervisor,
wherein the snapshot copy operation is performed by one of the one or more shared physical computer storage devices,
resulting in a snapshot of a given storage volume that uniquely corresponds to the selected one of the plurality of virtual machines,
wherein the metadata identifies the snapshot, and
storing the metadata in a database,
wherein the metadata that identifies the snapshot is associated with the selected one of the plurality of virtual machines;
a processor to execute the instructions; and
wherein, based on the command issued by the processor of the virtual server agent, the snapshot copy operation is performed for the selected one of the plurality of virtual machines without performing the snapshot copy operation for any unselected virtual machine having a storage volume in the one or more shared physical computer storage devices.
2. The virtual server agent of claim 1, wherein the virtual server agent is external to the hypervisor and is implemented on separate hardware.

64

3. The virtual server agent of claim 1, wherein the snapshot copy comprises virtual machine data stored in its native format.

4. The virtual server agent of claim 1, wherein the metadata comprises a snapshot identifier.

5. The virtual server agent of claim 1, wherein the instructions further cause the processor to issue a command to the hypervisor to perform a backup copy operation using the metadata corresponding to the completed snapshot copy operation.

6. The virtual server agent of claim 5, wherein the backup copy operation causes a backup copy of the virtual machine to be stored on one or more external computer storage devices.

7. The virtual server agent of claim 5, wherein the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

8. A virtual server agent method comprising:
by the virtual server agent, issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines,

- wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, and

- wherein each storage volume uniquely corresponds to one of the virtual machines in the plurality of virtual machines;

- by the virtual server agent, receiving metadata corresponding to the completed snapshot copy operation from the hypervisor,

- wherein the snapshot copy operation is performed by one of the one or more shared physical computer storage devices,

- resulting in a snapshot of a given storage volume that uniquely corresponds to the selected one of the plurality of virtual machines,

- wherein the metadata identifies the snapshot;

- by the virtual server agent, storing the metadata in a database,

- wherein the metadata that identifies the snapshot is associated with the selected one of the plurality of virtual machines; and

- wherein based on the command issued by the virtual server agent to the hypervisor, the snapshot copy operation is performed for the selected one of the plurality of virtual machines without performing the snapshot copy operation for any unselected virtual machine having a storage volume in the one or more shared physical computer storage devices.

9. The method of claim 8, wherein the snapshot copy comprises virtual machine data stored in its native format.

10. The method of claim 8, wherein the metadata comprises a snapshot identifier.

11. The method of claim 8, further comprising issuing a command to the hypervisor to perform a backup copy operation using the metadata corresponding to the completed snapshot copy operation.

12. The method of claim 11, wherein the backup copy operation causes a backup copy of the virtual machine to be stored on one or more external computer storage devices.

65

13. The method of claim 11, wherein the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

14. A non-transitory computer readable medium storing instructions which, when executed by at least one computing device, perform a virtual server agent method comprising:

- by the virtual server agent, issuing a command to a hypervisor to perform a snapshot copy operation for a selected one of a plurality of virtual machines operated by the hypervisor, the hypervisor being communicatively coupled to one or more shared physical computer storage devices which store the plurality of virtual machines, wherein a plurality of storage volumes are provided in the one or more shared physical computer storage devices, and wherein each storage volume uniquely corresponds to one of the virtual machines in the plurality of virtual machines;
- by the virtual server agent, receiving metadata corresponding to the completed snapshot copy operation from the hypervisor, wherein the snapshot copy operation is performed by one of the one or more physical computer storage devices, resulting in a snapshot of a given storage volume that uniquely corresponds to the selected one of the plurality of virtual machines, and wherein the metadata identifies the snapshot;
- by the virtual server agent, storing the metadata in a database,

66

wherein the metadata that identifies the snapshot is associated with the selected one of the plurality of virtual machines; and

wherein based on the command issued by the virtual server agent to the hypervisor, the snapshot copy operation is performed for the selected one of the plurality of virtual machines without performing the snapshot copy operation for any unselected virtual machine having a storage volume in the one or more shared physical computer storage devices.

15. The non-transitory computer readable medium of claim 14, wherein the snapshot copy comprises virtual machine data stored in its native format.

16. The non-transitory computer readable medium of claim 14, wherein the metadata comprises a snapshot identifier.

17. The non-transitory computer readable medium of claim 14, wherein the instructions cause the method to be performed by a virtual machine operated by the hypervisor.

18. The non-transitory computer readable medium of claim 14, wherein the instructions further cause the computing device to issue a command to the hypervisor to perform a backup copy operation using the metadata corresponding to the completed snapshot copy operation.

19. The non-transitory computer readable medium of claim 18, wherein the backup copy operation causes a backup copy of the virtual machine to be stored on one or more external computer storage devices.

20. The non-transitory computer readable medium of claim 18, wherein the backup copy comprises virtual machine data stored in a backup format which is different from a native format of the virtual machine data.

* * * * *

EXHIBIT F

(12) **United States Patent**
Prahlad et al.

(10) **Patent No.:** **US 10,248,657 B2**
(45) **Date of Patent:** **Apr. 2, 2019**

(54) **DATA OBJECT STORE AND SERVER FOR A CLOUD STORAGE ENVIRONMENT, INCLUDING DATA DEDUPLICATION AND DATA MANAGEMENT ACROSS MULTIPLE CLOUD STORAGE SITES**

(58) **Field of Classification Search**
CPC combination set(s) only.
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,686,620 A 8/1987 Ng
4,995,035 A 2/1991 Cole et al.
(Continued)

FOREIGN PATENT DOCUMENTS

CN 1259733 A 7/2000
CN 1525272 A 9/2004
(Continued)

OTHER PUBLICATIONS

"Amazon Web Services@Amazon.com." <http://web.archive.org/web/20080819144354/http://aws.amazon.com/s3>, Aug. 19, 2008.
(Continued)

Primary Examiner — Etienne P Leroux

(74) *Attorney, Agent, or Firm* — Perkins Coie LLP

(57) **ABSTRACT**

Data storage operations, including content-indexing, containerized deduplication, and policy-driven storage, are performed within a cloud environment. The systems support a variety of clients and cloud storage sites that may connect to the system in a cloud environment that requires data transfer over wide area networks, such as the Internet, which may have appreciable latency and/or packet loss, using various network protocols, including HTTP and FTP. Methods are disclosed for content indexing data stored within a cloud environment to facilitate later searching, including collaborative searching. Methods are also disclosed for performing containerized deduplication to reduce the strain on a system namespace, effectuate cost savings, etc. Methods are disclosed for identifying suitable storage locations, including suitable cloud storage sites, for data files subject to a storage policy. Further, systems and methods for providing a cloud

(Continued)

(71) Applicant: **Commvault Systems, Inc.**, Tinton Falls, NJ (US)

(72) Inventors: **Anand Prahlad**, Bangalore (IN);
Marcus S. Muller, Maynard, MA (US);
Rajiv Kottomtharayil, Marlboro, NJ (US);
Srinivas Kavuri, San Jose, CA (US);
Parag Gokhale, Marlboro, NJ (US);
Manoj Kumar Vijayan, Marlboro, NJ (US)

(73) Assignee: **Commvault Systems, Inc.**, Tinton Falls, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **15/258,252**

(22) Filed: **Sep. 7, 2016**

(65) **Prior Publication Data**

US 2017/0039218 A1 Feb. 9, 2017

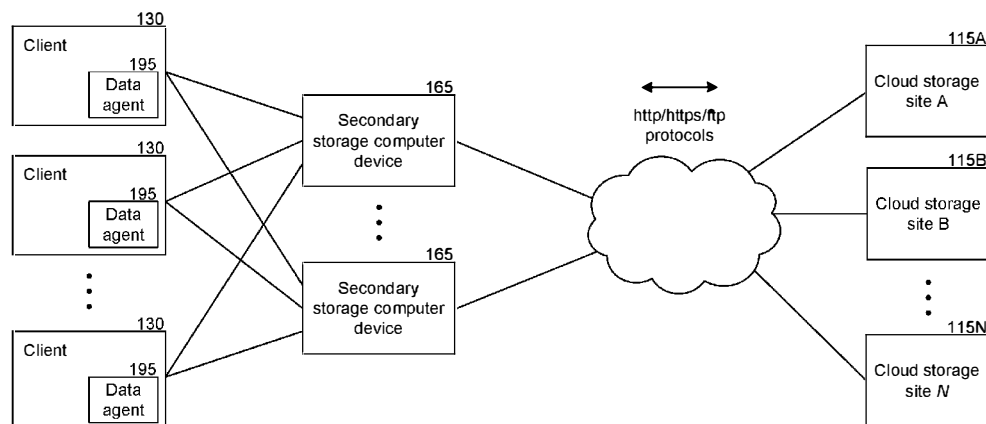
Related U.S. Application Data

(60) Division of application No. 14/494,674, filed on Sep. 24, 2014, now Pat. No. 9,454,537, which is a
(Continued)

(51) **Int. Cl.**
G06F 17/30 (2006.01)
G06Q 30/02 (2012.01)

(Continued)

(52) **U.S. Cl.**
CPC **G06F 17/30156** (2013.01); **G06F 3/0605** (2013.01); **G06F 3/067** (2013.01);
(Continued)



US 10,248,657 B2

Page 2

gateway and a scalable data object store within a cloud environment are disclosed, along with other features.

23 Claims, 33 Drawing Sheets

Related U.S. Application Data

continuation of application No. 13/615,999, filed on Sep. 14, 2012, now Pat. No. 8,849,761, which is a continuation of application No. 12/751,850, filed on Mar. 31, 2010, now Pat. No. 8,285,681.

(60) Provisional application No. 61/299,313, filed on Jan. 28, 2010, provisional application No. 61/223,695, filed on Jul. 7, 2009, provisional application No. 61/221,993, filed on Jun. 30, 2009.

(51) Int. Cl.

G06Q 50/18 (2012.01)
H04L 29/06 (2006.01)
G06F 3/06 (2006.01)
H04L 29/08 (2006.01)
G06F 11/34 (2006.01)

(52) U.S. Cl.

CPC *G06F 3/0631* (2013.01); *G06F 3/0667* (2013.01); *G06F 17/302* (2013.01); *G06F 17/3002* (2013.01); *G06F 17/30082* (2013.01); *G06F 17/30215* (2013.01); *G06Q 30/02* (2013.01); *G06Q 30/0206* (2013.01); *G06Q 50/188* (2013.01); *H04L 63/0428* (2013.01); *H04L 67/1095* (2013.01); *H04L 67/1097* (2013.01); *H04L 67/28* (2013.01); *H04L 67/2852* (2013.01); *G06F 11/3485* (2013.01); *G06F 2003/0697* (2013.01); *H04L 67/02* (2013.01); *H04L 67/06* (2013.01); *H04L 67/22* (2013.01); *H04L 69/08* (2013.01)

(56) References Cited

U.S. PATENT DOCUMENTS

5,005,122 A 4/1991 Griffin et al.
 5,093,912 A 3/1992 Dong et al.
 5,123,107 A 6/1992 Mensch, Jr.
 5,133,065 A 7/1992 Cheffetz et al.
 5,193,154 A 3/1993 Kitajima et al.
 5,212,772 A 5/1993 Masters
 5,226,157 A 7/1993 Nakano et al.
 5,239,647 A 8/1993 Anglin et al.
 5,241,668 A 8/1993 Eastridge et al.
 5,241,670 A 8/1993 Eastridge et al.
 5,276,860 A 1/1994 Fortier et al.
 5,276,867 A 1/1994 Kenley et al.
 5,287,500 A 2/1994 Stoppani, Jr.
 5,321,816 A 6/1994 Rogan et al.
 5,333,315 A 7/1994 Saether et al.
 5,347,653 A 9/1994 Flynn et al.
 5,410,700 A 4/1995 Fecteau et al.
 5,448,724 A 9/1995 Hayashi
 5,491,810 A 2/1996 Allen
 5,495,607 A 2/1996 Pisello et al.
 5,504,873 A 4/1996 Martin et al.
 5,544,345 A 8/1996 Carpenter et al.
 5,544,347 A 8/1996 Yanai et al.
 5,559,957 A 9/1996 Balk
 5,619,644 A 4/1997 Crockett et al.
 5,638,509 A 6/1997 Dunphy et al.
 5,664,204 A 9/1997 Wang
 5,673,381 A 9/1997 Huai et al.
 5,699,361 A 12/1997 Ding et al.

5,729,743 A 3/1998 Squibb
 5,751,997 A 5/1998 Kullick et al.
 5,758,359 A 5/1998 Saxon
 5,761,677 A 6/1998 Senator et al.
 5,764,972 A 6/1998 Crouse et al.
 5,778,395 A 7/1998 Whiting et al.
 5,812,398 A 9/1998 Nielsen
 5,813,009 A 9/1998 Johnson et al.
 5,813,017 A 9/1998 Morris
 5,875,478 A 2/1999 Blumenau
 5,887,134 A 3/1999 Ebrahim
 5,901,327 A 5/1999 Ofek
 5,924,102 A 7/1999 Perks
 5,950,205 A 9/1999 Aviani, Jr.
 5,974,563 A 10/1999 Beeler, Jr.
 6,021,415 A 2/2000 Cannon et al.
 6,026,414 A 2/2000 Anglin
 6,052,735 A 4/2000 Ulrich et al.
 6,076,148 A 6/2000 Kedem
 6,094,416 A 7/2000 Ying
 6,131,095 A 10/2000 Low et al.
 6,131,190 A 10/2000 Sidwell
 6,148,412 A 11/2000 Cannon et al.
 6,154,787 A 11/2000 Urevig et al.
 6,161,111 A 12/2000 Mutalik et al.
 6,167,402 A 12/2000 Yeager
 6,212,512 B1 4/2001 Barney et al.
 6,260,069 B1 7/2001 Anglin
 6,269,431 B1 7/2001 Dunham
 6,275,953 B1 8/2001 Vahalia et al.
 6,301,592 B1 10/2001 Aoyama et al.
 6,324,581 B1 11/2001 Xu et al.
 6,327,590 B1 12/2001 Chidlovskii et al.
 6,328,766 B1 12/2001 Long
 6,330,570 B1 12/2001 Crighton
 6,330,642 B1 12/2001 Carteau
 6,343,324 B1 1/2002 Hubis et al.
 RE37,601 E 3/2002 Eastridge et al.
 6,356,801 B1 3/2002 Goodman et al.
 6,389,432 B1 5/2002 Pothapragada et al.
 6,421,711 B1 7/2002 Blumenau et al.
 6,487,561 B1 11/2002 Ofek et al.
 6,519,679 B2 2/2003 Devireddy et al.
 6,538,669 B1 3/2003 Lagueux, Jr. et al.
 6,564,228 B1 5/2003 O'Connor
 6,658,526 B2 12/2003 Nguyen et al.
 6,941,429 B1 9/2005 Kamvyselis et al.
 6,959,327 B1 * 10/2005 Vogl H04L 47/26 370/229
 6,973,555 B2 12/2005 Fujiwara et al.
 7,000,238 B2 * 2/2006 Nadler G06F 9/548 709/217
 7,035,880 B1 4/2006 Crescenti et al.
 7,079,341 B2 7/2006 Kistler et al.
 7,096,418 B1 * 8/2006 Singhal G06F 17/30902 707/E17.12
 7,107,298 B2 9/2006 Prahlad et al.
 7,130,272 B1 10/2006 Gai et al.
 7,225,220 B2 5/2007 Gonzalez et al.
 7,246,207 B2 7/2007 Kottomtharayil et al.
 7,260,633 B2 8/2007 Lette et al.
 7,340,616 B2 3/2008 Rothman et al.
 7,343,356 B2 3/2008 Prahlad et al.
 7,343,453 B2 3/2008 Prahlad et al.
 7,346,751 B2 3/2008 Prahlad et al.
 7,366,846 B2 4/2008 Boyd et al.
 7,386,744 B2 6/2008 Barr et al.
 7,395,282 B1 7/2008 Crescenti et al.
 7,448,079 B2 11/2008 Tremain
 7,472,079 B2 12/2008 Fellenstein et al.
 7,483,895 B2 * 1/2009 Hysom G06F 17/30011
 7,502,820 B2 3/2009 Manders et al.
 7,516,346 B2 4/2009 Pinheiro et al.
 7,526,798 B2 4/2009 Chao et al.
 7,546,475 B2 6/2009 Mayo et al.
 7,587,570 B2 9/2009 Sarkar et al.
 7,617,191 B2 11/2009 Wilbrink et al.
 7,627,827 B2 12/2009 Taylor et al.
 7,631,351 B2 12/2009 Erofeev

US 10,248,657 B2

Page 3

(56)	References Cited			2005/0289414 A1 *	12/2005	Adya	G06F 11/1441
U.S. PATENT DOCUMENTS				2006/0058994 A1	3/2006	Ravi et al.	714/724
7,651,593 B2	1/2010	Prahlad et al.		2006/0101174 A1	5/2006	Kanamaru et al.	
7,653,668 B1 *	1/2010	Shelat	G06F 11/2094	2006/0190775 A1	8/2006	Aggarwal et al.	
			707/610	2006/0206507 A1	9/2006	Dahbour	
				2006/0224846 A1	10/2006	Amarendran et al.	
7,668,884 B2	2/2010	Prahlad et al.		2006/0236073 A1 *	10/2006	Soules	G06F 3/0613
7,685,269 B1	3/2010	Thrasher et al.					711/216
7,694,070 B2	4/2010	Mogi et al.		2006/0242356 A1	10/2006	Mogi et al.	
7,739,548 B2	6/2010	Goodrum et al.		2006/0245411 A1	11/2006	Chen et al.	
7,761,736 B2	7/2010	Nguyen et al.		2006/0251067 A1 *	11/2006	DeSanti	H04L 29/12801
7,765,167 B2	7/2010	Prahlad et al.					370/389
7,769,616 B2	8/2010	Ollivier		2007/0079156 A1	4/2007	Fujimoto	
7,778,984 B2	8/2010	Zhang et al.		2007/0101173 A1	5/2007	Fung	
7,792,789 B2	9/2010	Prahlad et al.		2007/0168606 A1	7/2007	Takai et al.	
7,797,453 B2	9/2010	Meijer et al.		2008/0005168 A1	1/2008	Huff et al.	
7,814,149 B1	10/2010	Stringham		2008/0162592 A1	7/2008	Huang et al.	
7,814,351 B2	10/2010	Lubbers et al.		2008/0183891 A1	7/2008	Ni et al.	
7,818,082 B2	10/2010	Roumeliotis et al.		2008/0228771 A1	9/2008	Prahlad et al.	
7,822,967 B2	10/2010	Fung		2008/0244032 A1	10/2008	Gilson et al.	
7,840,537 B2	11/2010	Gokhale et al.		2008/0256384 A1	10/2008	Branson et al.	
7,882,077 B2	2/2011	Gokhale et al.		2008/0270461 A1 *	10/2008	Gordon	G06F 3/0608
7,899,788 B2	3/2011	Chandhok et al.		2009/0198677 A1	8/2009	Sheehy et al.	
7,917,438 B2	3/2011	Kenedy et al.		2009/0198825 A1 *	8/2009	Miller	H04L 67/104
7,996,270 B2	8/2011	Sundaresan					709/230
8,001,277 B2	8/2011	Mega et al.		2009/0210464 A1	8/2009	Chiang-Lin	
8,037,028 B2	10/2011	Prahlad et al.		2009/0268903 A1	10/2009	Bojinov et al.	
8,065,166 B2	11/2011	Maresh et al.		2009/0282020 A1 *	11/2009	McSheffrey	G06F 17/3002
8,108,427 B2	1/2012	Prahlad et al.		2009/0287665 A1	11/2009	Prahlad et al.	
8,112,605 B2	2/2012	Kavuri		2009/0319534 A1	12/2009	Gokhale	
8,134,727 B1	3/2012	Shmunis et al.		2010/0023722 A1 *	1/2010	Tabbara	H04L 43/00
8,140,786 B2	3/2012	Bunte et al.					711/170
8,219,524 B2	7/2012	Gokhale		2010/0064033 A1	3/2010	Travostino et al.	
8,266,406 B2	9/2012	Kavuri		2010/0070448 A1	3/2010	Omoigui	
8,285,681 B2	10/2012	Prahlad et al.		2010/0070466 A1	3/2010	Prahlad et al.	
8,296,534 B1	10/2012	Gupta et al.		2010/0070474 A1	3/2010	Lad	
8,307,177 B2	11/2012	Prahlad et al.		2010/0082672 A1	4/2010	Kottomtharayil et al.	
8,316,091 B2	11/2012	Hirvela et al.		2010/0082700 A1	4/2010	Parab	
8,321,688 B2	11/2012	Auradkar et al.		2010/0082713 A1	4/2010	Frid-Nielsen et al.	
8,352,608 B1	1/2013	Keagy et al.		2010/0162002 A1	6/2010	Dodgson et al.	
8,364,802 B1	1/2013	Keagy et al.		2010/0235333 A1 *	9/2010	Bates	G06F 17/30156
8,370,307 B2	2/2013	Wolfe					707/692
8,396,838 B2	3/2013	Brockway et al.		2010/0257403 A1	10/2010	Virk et al.	
8,407,190 B2	3/2013	Prahlad et al.		2010/0269164 A1	10/2010	Sosnosky et al.	
8,417,697 B2	4/2013	Ghemawat et al.		2010/0274772 A1 *	10/2010	Samuels	G06F 17/30156
8,429,630 B2	4/2013	Nikolov et al.					707/693
8,434,131 B2	4/2013	Varadharajan et al.		2010/0318782 A1	12/2010	Auradkar et al.	
8,510,573 B2	8/2013	Muller et al.		2010/0332401 A1	12/2010	Prahlad et al.	
8,527,549 B2	9/2013	Cidon		2010/0333116 A1	12/2010	Prahlad et al.	
8,566,362 B2	10/2013	Mason, Jr. et al.		2011/0022642 A1	1/2011	deMilo et al.	
8,578,120 B2	11/2013	Attarde et al.		2011/0040824 A1	2/2011	Harm	
8,612,439 B2	12/2013	Prahlad et al.		2011/0191544 A1	8/2011	Naga et al.	
8,626,741 B2	1/2014	Vijayakumar et al.		2011/0277027 A1	11/2011	Hayton et al.	
8,635,184 B2	1/2014	Hsu et al.		2012/0054626 A1	3/2012	Odenheimer	
8,660,038 B1	2/2014	Pascasio et al.		2012/0110186 A1	5/2012	Kapur et al.	
8,674,823 B1	3/2014	Contario et al.		2012/0131645 A1	5/2012	Harm	
8,683,103 B2	3/2014	Ripberger		2012/0240183 A1	9/2012	Sinha	
8,707,070 B2	4/2014	Muller		2013/0007245 A1	1/2013	Malik et al.	
8,769,048 B2	7/2014	Kottomtharayil		2013/0035795 A1	2/2013	Pfeiffer et al.	
8,780,400 B2	7/2014	Shmunis et al.		2013/0125198 A1	5/2013	Ferguson et al.	
8,799,242 B2	8/2014	Leonard et al.		2013/0238572 A1	9/2013	Prahlad et al.	
8,849,761 B2	9/2014	Prahlad et al.		2013/0238969 A1	9/2013	Smith et al.	
8,849,955 B2	9/2014	Prahlad et al.		2013/0262385 A1	10/2013	Kumarasamy et al.	
8,924,511 B2	12/2014	Brand		2013/0297902 A1	11/2013	Collins et al.	
8,950,009 B2	2/2015	Vijayan et al.		2013/0326279 A1	12/2013	Chavda et al.	
9,021,282 B2	4/2015	Muller		2014/0189432 A1	7/2014	Gokhale et al.	
9,021,307 B1	4/2015	Parameswaran et al.		2014/0283010 A1	9/2014	Rutkowski et al.	
9,454,537 B2	9/2016	Prahlad et al.		2015/0113055 A1	4/2015	Vijayan et al.	
2002/0035511 A1	3/2002	Haji et al.		2015/0127967 A1	5/2015	Dutton et al.	
2002/0083079 A1	6/2002	Meier et al.		2015/0198995 A1	7/2015	Muller	
2002/0095609 A1	7/2002	Tokunaga		2016/0100013 A1	4/2016	Vijayan et al.	
2002/0129106 A1	9/2002	Gutfreund		2016/0224651 A1	8/2016	Kumarasamy et al.	
2002/0194033 A1	12/2002	Huff					
2002/0194511 A1	12/2002	Swoboda					
2003/0140068 A1	7/2003	Yeung					
2003/0200222 A1	10/2003	Feinberg et al.					
2004/0210724 A1	10/2004	Koning et al.					
2005/0076251 A1	4/2005	Barr et al.					
				CN	1936818	3/2007	
				CN	1968254	5/2007	
				FOREIGN PATENT DOCUMENTS			

US 10,248,657 B2

Page 4

(56) References Cited

FOREIGN PATENT DOCUMENTS

EP	0259912 A1	3/1988
EP	0405926 A2	1/1991
EP	0467546 A2	1/1992
EP	0809184 A1	11/1997
EP	0817040 A2	1/1998
EP	0899662 A1	3/1999
EP	0981090 A1	2/2000
W●	W●-9513580 A1	5/1995
W●	W●-9912098 A1	3/1999

OTHER PUBLICATIONS

Armstead et al., "Implementation of a Campwide Distributed Mass Storage Service: The Dream vs. Reality," IEEE, Sep. 11-14, 1995, pp. 190-199.

Arneson, "Mass Storage Archiving in Network Environments," Digest of Papers, Ninth IEEE Symposium on Mass Storage Systems, Oct. 31, 1988-Nov. 3, 1988, pp. 45-50, Monterey, CA.

Bates, S. et al., "Sharepoint 2007 User's Guide," pp. 1-88, 2007, Springer-Verlag New York, Inc., 104 pages.

Brandon, J., "Virtualization Shakes Up Backup Strategy," <http://www.computerworld.com>, internet accessed on Mar. 6, 2008, 3 pages.

Cabrera et al., "ADSM: A Multi-Platform, Scalable, Backup and Archive Mass Storage System," Digest of Papers, Compcon '95, Proceedings of the 40th IEEE Computer Society International Conference, Mar. 5, 1995-Mar. 9, 1995, pp. 420-427, San Francisco, CA.

Chiappetta, Marco, "ESA Enthusiast System Architecture," <http://hothardware.com/Articles/NVIDIA_ESA_Enthusiast_System_Architecture/>, Nov. 5, 2007, 2 pages.

CommVault Systems, Inc., "A CommVault White Paper: VMware Consolidated Backup (VCB) Certification Information Kit," 2007, 23 pages.

CommVault Systems, Inc., "CommVault Solutions—VMware," <http://www.commvault.com/solutions/vmware/>, internet accessed Mar. 24, 2008, 2 pages.

CommVault Systems, Inc., "Enhanced Protection and Manageability of Virtual Servers," Partner Solution Brief, 2008, 6 pages.

Davis, D., "3 VMware Consolidated Backup (VCB) Utilities You Should Know," Petri IT Knowledgebase, <http://www.petri.co.il/vmware-consolidated-backup-utilities.htm>, internet accessed on Jul. 14, 2008, Jan. 7, 2008.

Davis, D., "Understanding VMware VMX Configuration Files," Petri IT Knowledgebase, <http://www.petri.co.il/virtual_vmware_vmx_configuration_files.htm>, internet accessed on Jun. 19, 2008, 6 pages.

Davis, D., "VMware Server & Workstation Disk Files Explained," Petri IT Knowledgebase, <http://www.petri.co.il/virtual_vmware_files_explained.htm>, internet accessed on Jun. 19, 2008, 5 pages.

Davis, D., "VMware Versions Compared," Petri IT Knowledgebase, <http://www.petri.co.il/virtual_vmware_versions_compared.htm>, internet accessed on Apr. 28, 2008, 6 pages.

Eitel, "Backup and Storage Management in Distributed Heterogeneous Environments," IEEE, Jun. 12-16, 1994, pp. 124-126.

Extended European Search Report for Application No. 08798909.1, dated Mar. 21, 2016, 11 pages.

Foster, et al., "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop, 2008.

Gait, J., "The Optical File Cabinet: A Random-Access File System for Write-Once Optical Disks," IEEE Computer, vol. 21, No. 6, pp. 11-22 (Jun. 1988).

International Search Report and Written Opinion for International Application No. PCT/US08/74686, dated Jun. 22, 2009, 13 pages.

International Search Report and Written Opinion for International Application No. PCT/US2010/040402, dated Feb. 24, 2011, 11 pages.

International Search Report and Written Opinion for PCT/US2011/054374, dated May 2, 2012, 9 pages.

Jander, M., "Launching Storage-Area Net," Data Communications, US, McGraw Hill, NY, vol. 27, No. 4 (Mar. 21, 1998), pp. 64-72.

Liu et al., "Semantic data de-duplication for archival storage system", IEEE 2008, 9 pages.

Mell et al., "The NIST Definition of Cloud Computing," NIST Special Publication, 2011.

Microsoft Corporation, "How NTFS Works," Windows Server TechCenter, updated Mar. 28, 2003, Internet accessed Mar. 26, 2008, 26 pages.

Parlante, Nick; Linked List Basics; 2001, pp. 1-26.

Rosenblum et al., "The Design and Implementation of a Log-Structured File System," Operating Systems Review SIGOPS, vol. 25, No. 5, New York, US, pp. 1-15 (May 1991).

Sanbarrow.com, "Disktype-table," <http://sanbarrow.com/vmdk/disktypes.html>, internet accessed on Jul. 22, 2008, 4 pages.

Sanbarrow.com, "Files Used by a VM," <http://sanbarrow.com/vmx/vmx-files-used-by-a-vm.html>, internet accessed on Jul. 22, 2008, 2 pages.

Sanbarrow.com, "Monolithic Versus Split Disks," <http://sanbarrow.com/vmdk/monolithicversussplit.html>, internet accessed on Jul. 14, 2008, 2 pages.

VMware, Inc., "Open Virtual Machine Format," <http://www.vmware.com/appliances/learn/ovf.html>, internet accessed on May 6, 2008, 2 pages.

VMware, Inc., "OVF, Open Virtual Machine Format Specification, version 0.9," White Paper, <http://www.vmware.com>, 2007, 50 pages.

VMware, Inc., "The Open Virtual Machine Format Whitepaper for OVF Specification, version 0.9," White Paper, <http://www.vmware.com>, 2007, 16 pages.

VMware, Inc., "Understanding VMware Consolidated Backup," White Paper, <http://www.vmware.com>, 2007, 11 pages.

VMware, Inc., "Using VMware Infrastructure for Backup and Restore," Best Practices, <http://www.vmware.com>, 2006, 20 pages.

VMware, Inc., "Virtual Disk API Programming Guide," <http://www.vmware.com>, Revision Apr. 11, 2008, 2008, 44 pages.

VMware, Inc., "Virtual Disk Format 1.1," VMware Technical Note, <http://www.vmware.com>, Revision Nov. 13, 2007, Version 1.1, 2007, 18 pages.

VMware, Inc., "Virtual Machine Backup Guide, ESX Server 3.0.1 and VirtualCenter 2.0.1," <http://www.vmware.com>, updated Nov. 21, 2007, 74 pages.

VMware, Inc., "Virtual Machine Backup Guide, ESX Server 3.5, ESX Server 3i version 3.5, VirtualCenter 2.5," <http://www.vmware.com>, updated Feb. 21, 2008, 78 pages.

VMware, Inc., "Virtualized iSCSI SANs: Flexible, Scalable Enterprise Storage for Virtual Infrastructures," White Paper, <http://www.vmware.com>, Mar. 2008, 13 pages.

VMware, Inc., "VMware Consolidated Backup, Improvements in Version 3.5," Information Guide, <http://www.vmware.com>, 2007, 11 pages.

VMware, Inc., "VMware Consolidated Backup," Product Datasheet, <http://www.vmware.com>, 2007, 2 pages.

VMware, Inc., "VMware ESX 3.5," Product Datasheet, <http://www.vmware.com>, 2008, 4 pages.

VMware, Inc., "VMware GSX Server 3.2, Disk Types: Virtual and Physical," <http://www.vmware.com/support/gsx3/doc/disk_types_gsx.html>, internet accessed on Mar. 25, 2008, 2 pages.

VMware, Inc., "VMware OVF Tool," Technical Note, <http://www.vmware.com>, 2007, 4 pages.

VMware, Inc., "VMware Workstation 5.0, Snapshots in a Linear Process," <http://www.vmware.com/support/ws5/doc/ws_preserve_sshot_linear.html>, internet accessed on Mar. 25, 2008, 1 page.

VMware, Inc., "VMware Workstation 5.0, Snapshots in a Process Tree," <http://www.vmware.com/support/ws5/doc/ws_preserve_sshot_tree.html>, internet accessed on Mar. 25, 2008, 1 page.

VMware, Inc., "VMware Workstation 5.5, What Files Make Up a Virtual Machine?" <http://www.vmware.com/support/ws55/doc/ws_learning_files_in_a_vm.html>, internet accessed on Mar. 25, 2008, 2 pages.

Wang et al. "Performance analysis of data deduplication technology for storage", Proc. of SPIE vol. 7517, 2007, 7 pages.

US 10,248,657 B2

Page 5

(56)

References Cited**OTHER PUBLICATIONS**

Wikipedia, "Cloud computing," <http://en.wikipedia.org/wiki/Cloud_computing>, internet accessed Jul. 8, 2009, 13 pages.

Wikipedia, "Cluster (file system)," <http://en.wikipedia.org/wiki/Cluster_%28file_system%29>, internet accessed Jul. 25, 2008, 1 page.

Wikipedia, "Cylinder-head-sector," <<http://en.wikipedia.org/wiki/Cylinder-head-sector>>, internet accessed Jul. 22, 2008, 6 pages.

Wikipedia, "File Allocation Table," <http://en.wikipedia.org/wiki/File_Allocation_Table>, internet accessed on Jul. 25, 2008, 19 pages.

Wikipedia, "Logical Disk Manager," <http://en.wikipedia.org/wiki/Logical_Disk_Manager>, internet accessed Mar. 26, 2008, 3 pages.

Wikipedia, "Logical Volume Management," <http://en.wikipedia.org/wiki/Logical_volume_management>, internet accessed on Mar. 26, 2008, 5 pages.

Wikipedia, "Storage Area Network," <http://en.wikipedia.org/wiki/Storage_area_network>, internet accessed on Oct. 24, 2008, 5 pages.

Wikipedia, "Virtualization," <<http://en.wikipedia.org/wiki/Virtualization>>, internet accessed Mar. 18, 2008, 7 pages.

Anonymous: "Data Deduplication technology review," ComputerWeekly, Oct. 2008, pp. 1-6.

Partial Supplementary European Search Report in European Patent Application No. 10794641.0, dated Dec. 6, 2016, 7 pages.

Strickland, J., "How Cloud Storage Works," Apr. 30, 2008, pp. 1-6, retrieved from the inter: <http://computer.howstuffworks.com/cloud-computing/cloud-storage.htm/printable>.

Extended European Search Report for Application No. 10794641.0, dated Mar. 14, 2017, 14 pages.

Examination Report dated Jun. 27, 2018 in European Patent Application No. 10794641.0, 8 pages.

European Patent Office, Examination Report dated Nov. 29, 2018 in European Patent Application No. 10794641.0, 8 pages.

* cited by examiner

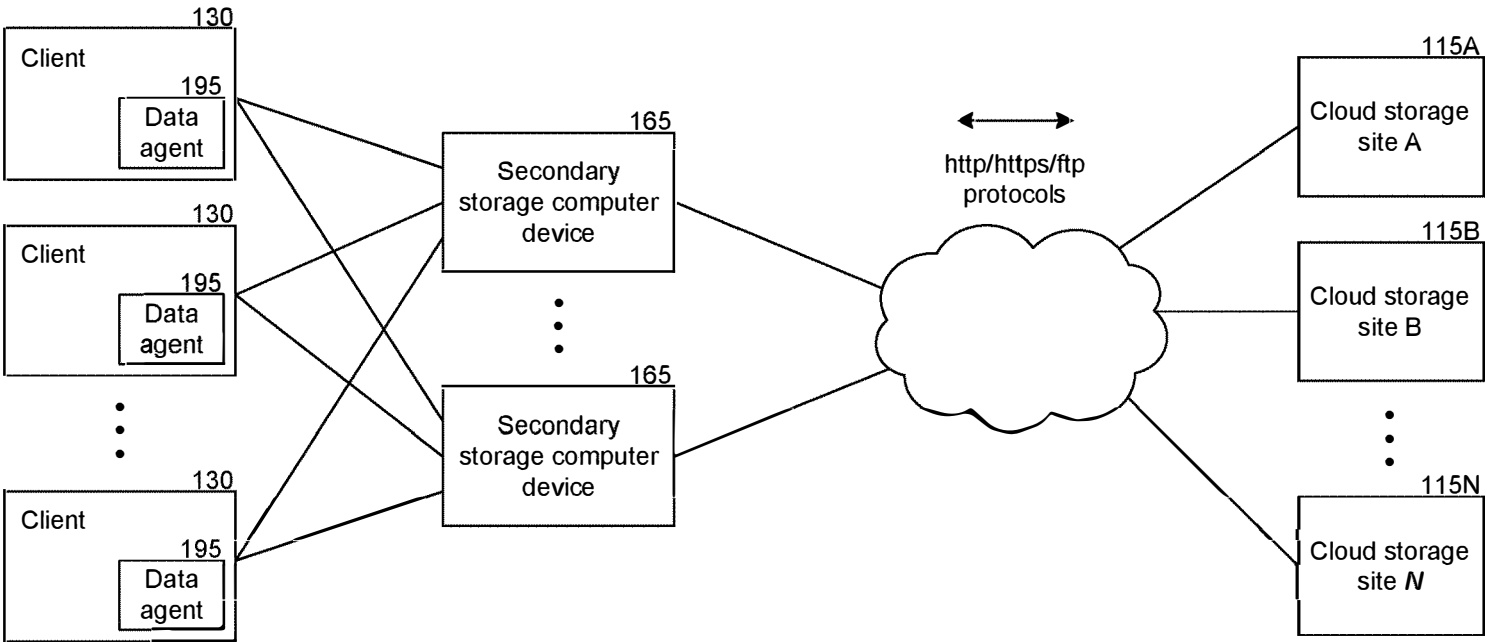


FIG. 1

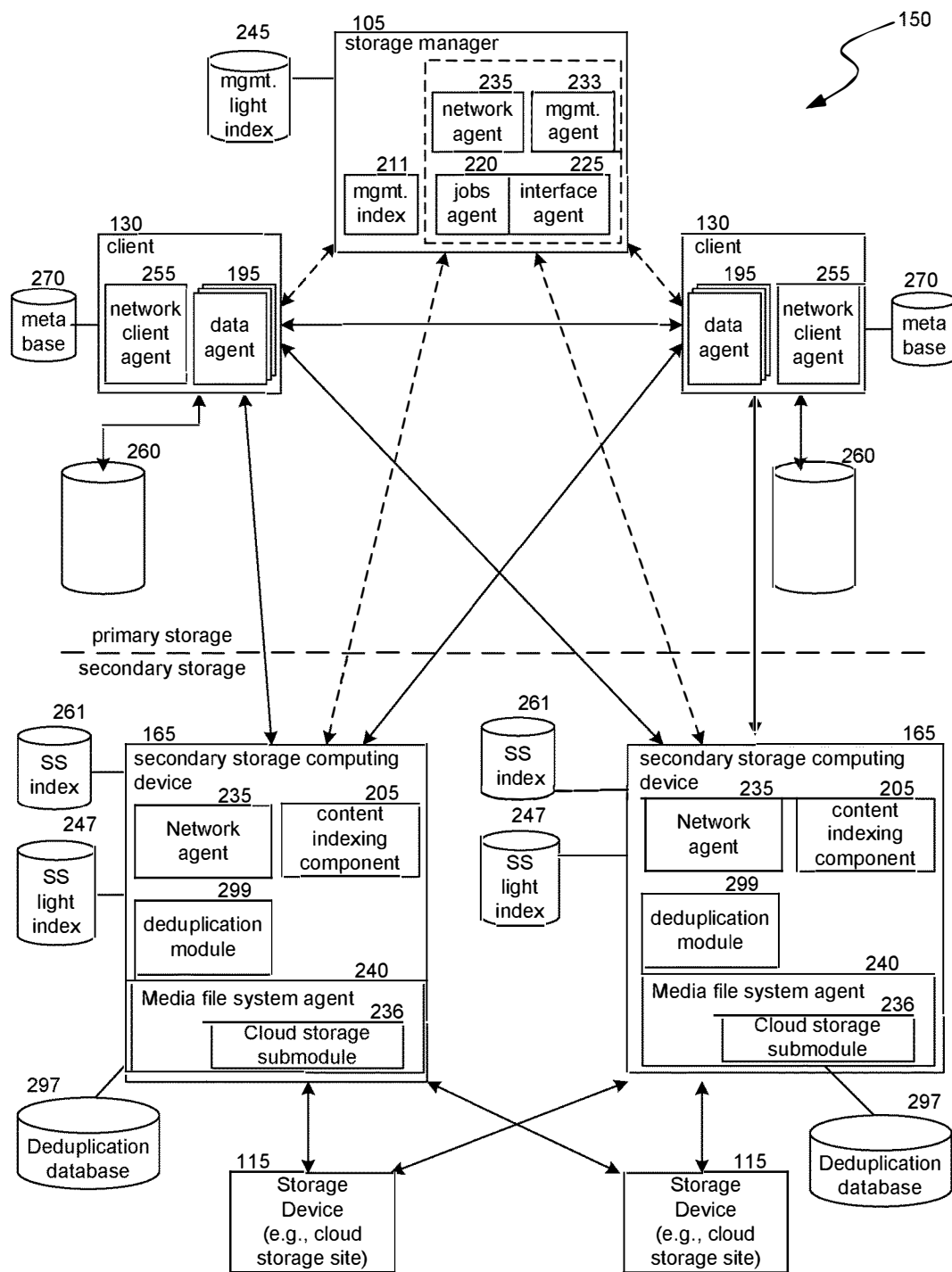
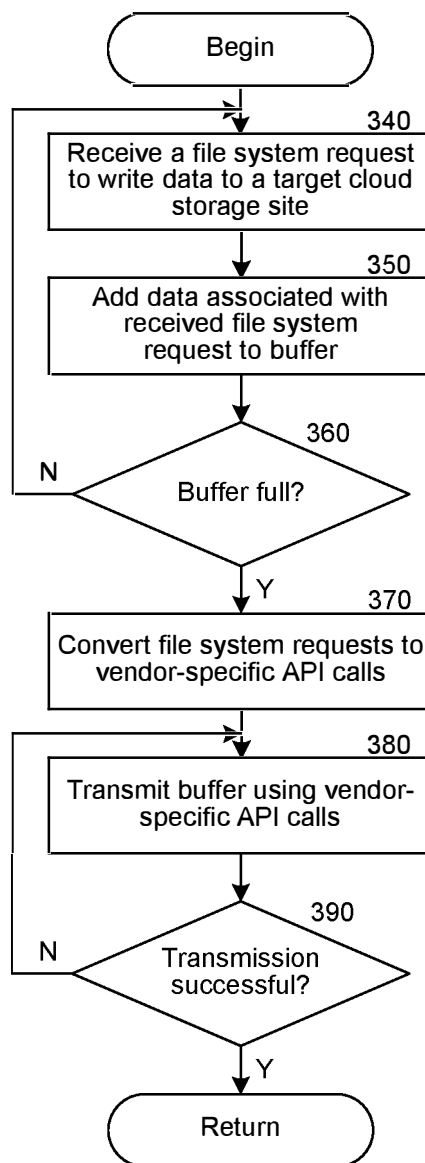


FIG. 2

**FIG. 3A**

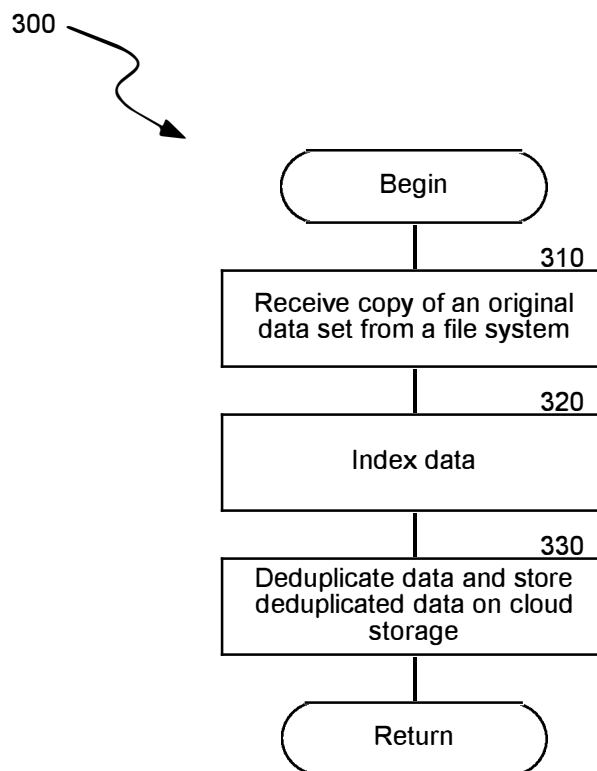


FIG. 3B

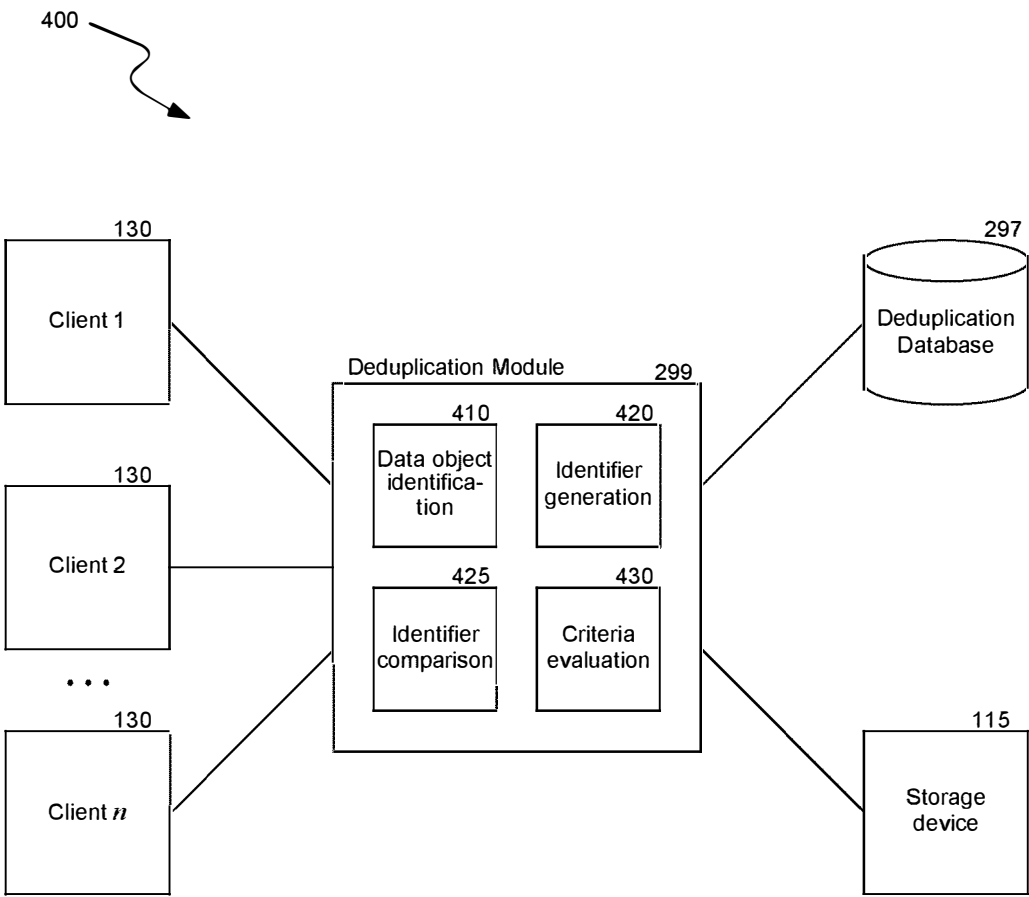
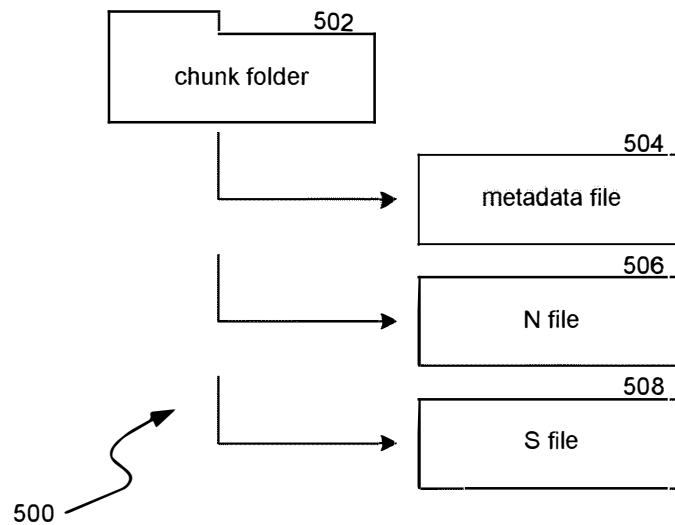
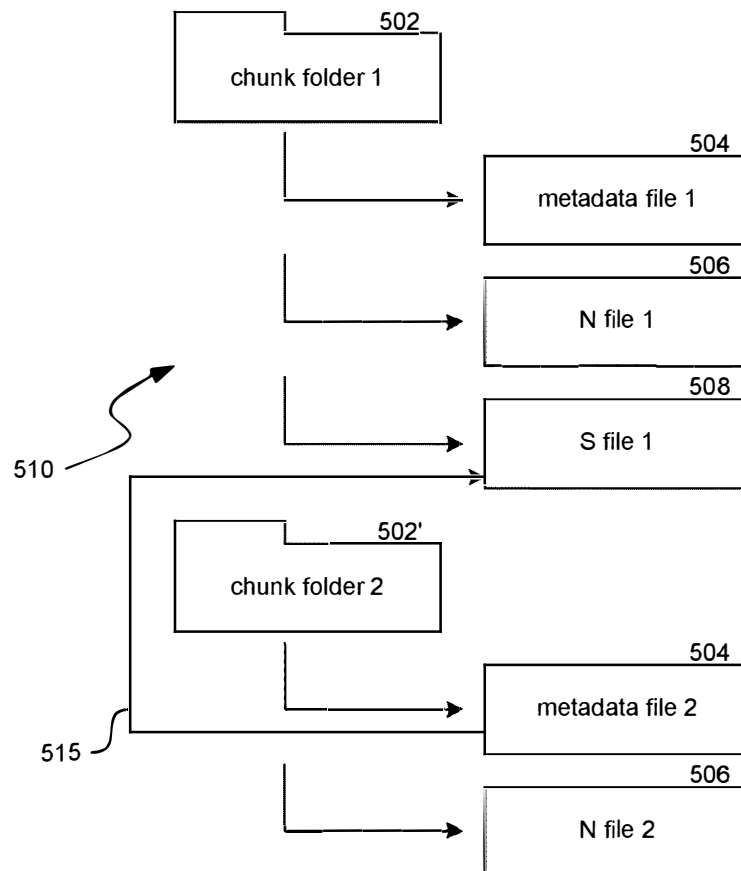


FIG. 4

**FIG. 5A****FIG. 5B**

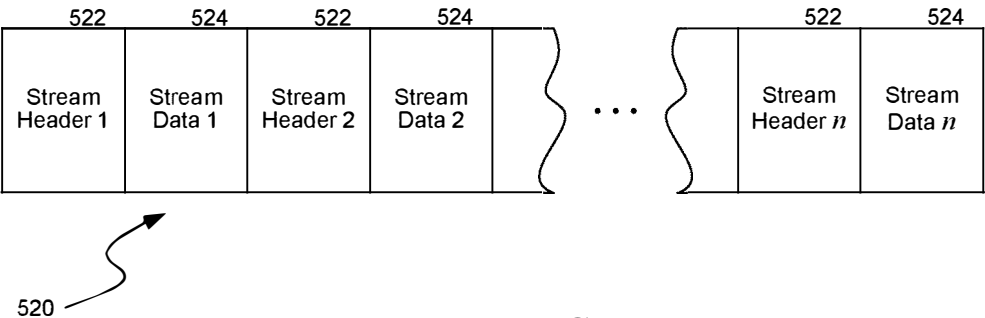


FIG. 5C

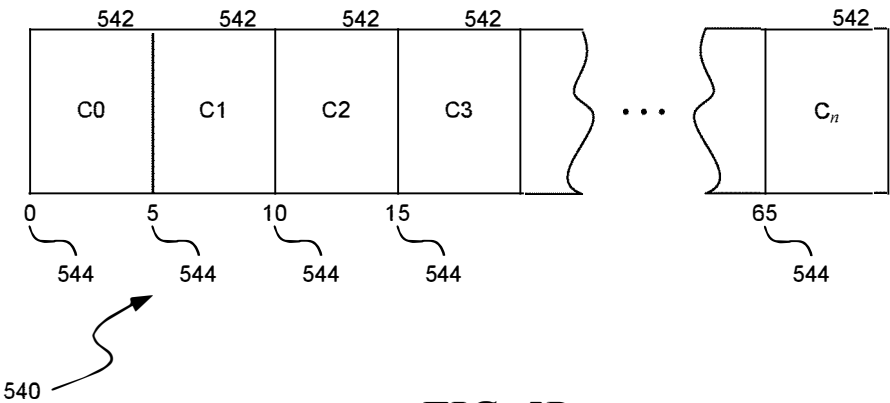
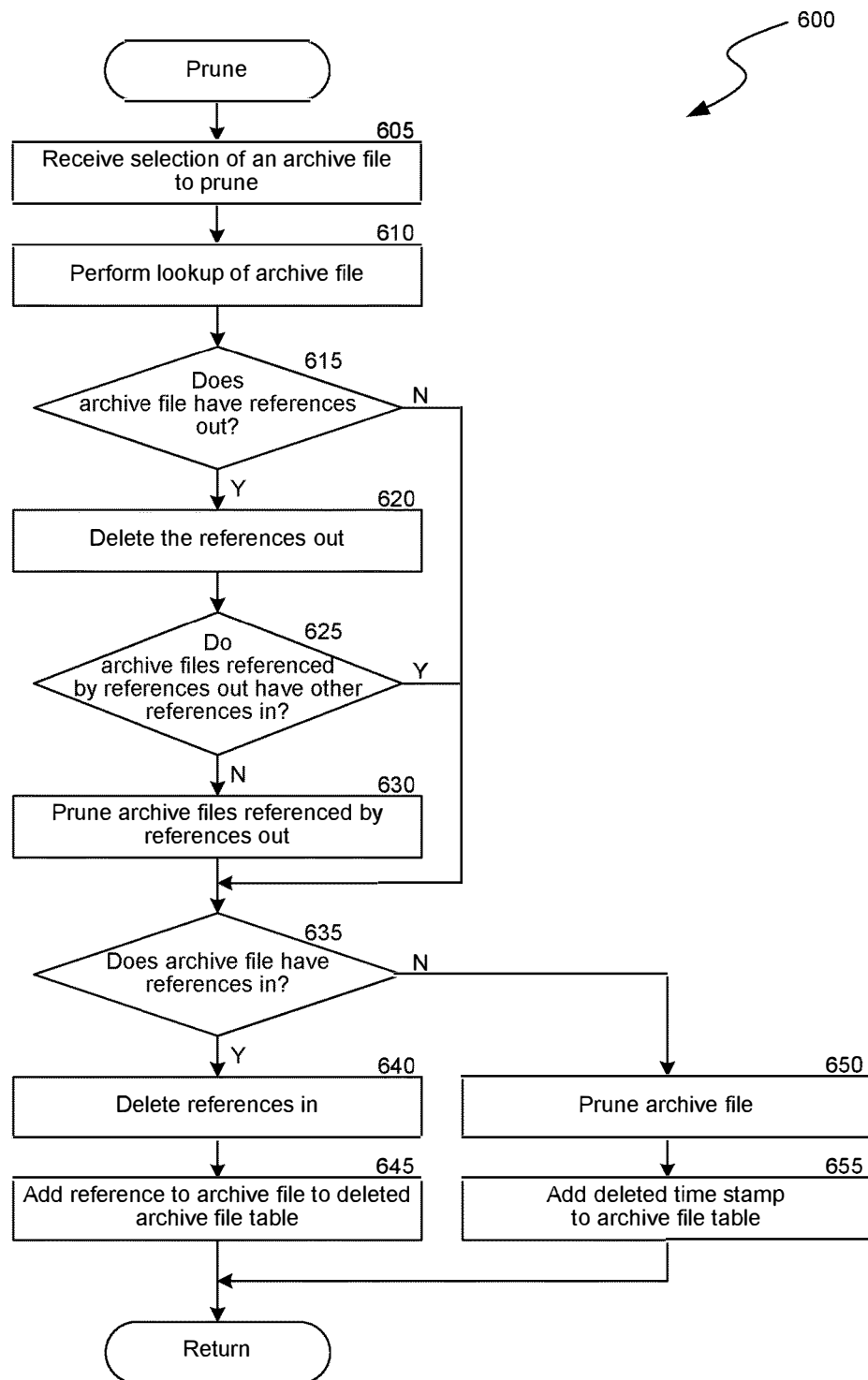


FIG. 5D

U.S. Patent**Apr. 2, 2019****Sheet 8 of 33****US 10,248,657 B2****FIG. 6**

700

710	720	730
Primary Rec. ID	File ID	Location
742 1	F ₁	AF ₀ , OF ₀
744 2	F ₁	AF ₁ , OF ₁
746 3	F ₃	AF ₁ , OF ₃
748 4	F ₃	AF ₃ , OF ₃
...

FIG. 7A

750

760	765	770	775	780
Secondary Rec. ID	Archive File ID	File	Ref _{IN}	Ref _{OUT}
792 1	AF ₀	F _T	AF _T , OF _T	- - - - -
794 2	AF ₁	F ₁		AF ₀ , OF ₀
796 3	AF ₁	F ₃	AF ₃ , OF ₃	
798 4	AF ₃	F ₃		AF ₁ , OF ₃
...

FIG. 7B

	754	756	758	762	752
	Rec. ID	Archive File ID	Ref _I N	Deleted Timestamp	
772	1	AF ₀	AF ₁ , OF ₁	T ₁	
774	2	AF ₁	AF ₃ , OF ₃	T ₂	
776	3	AF ₃	_____	T ₂	
	

FIG. 7C

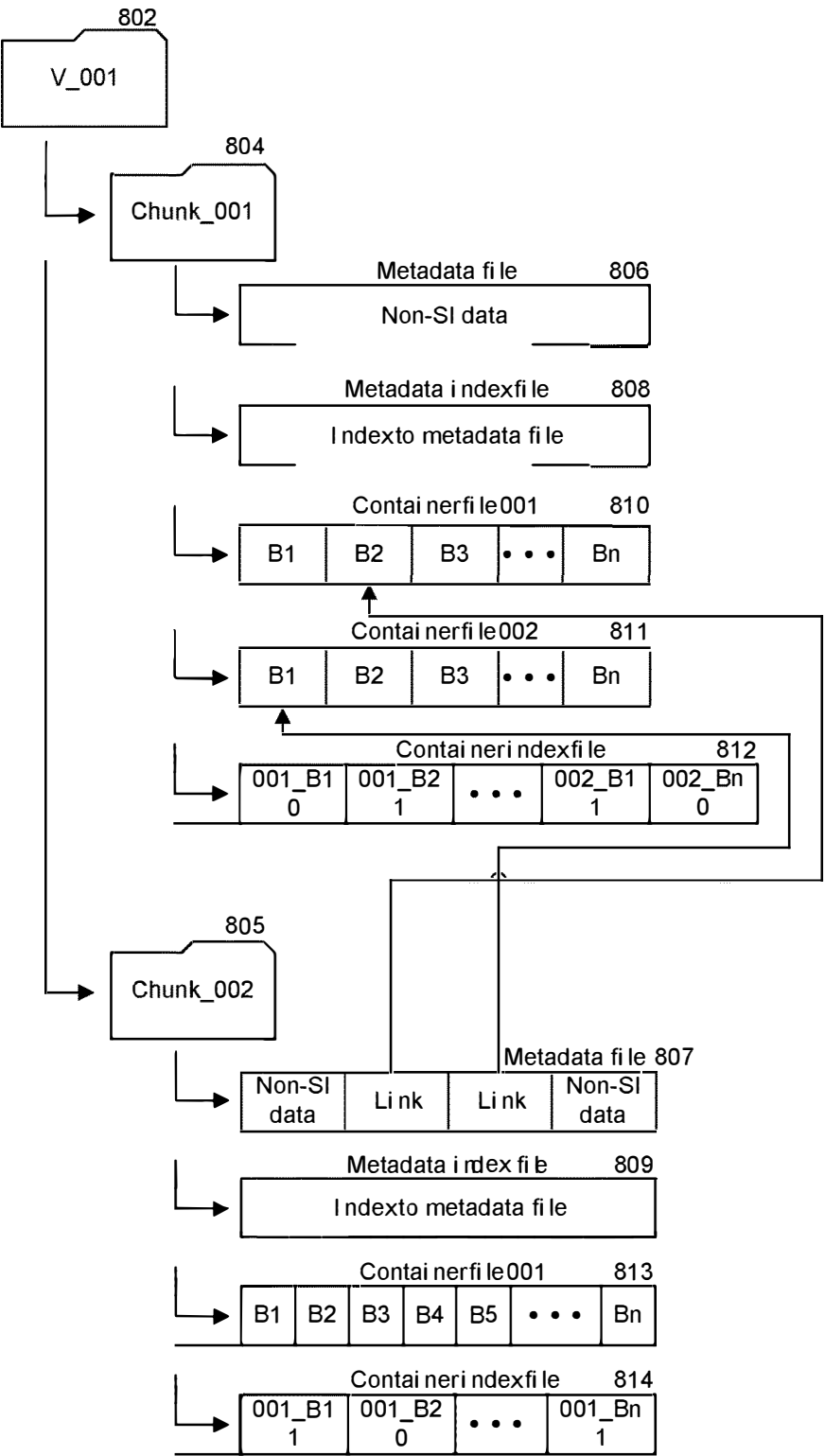


FIG. 8

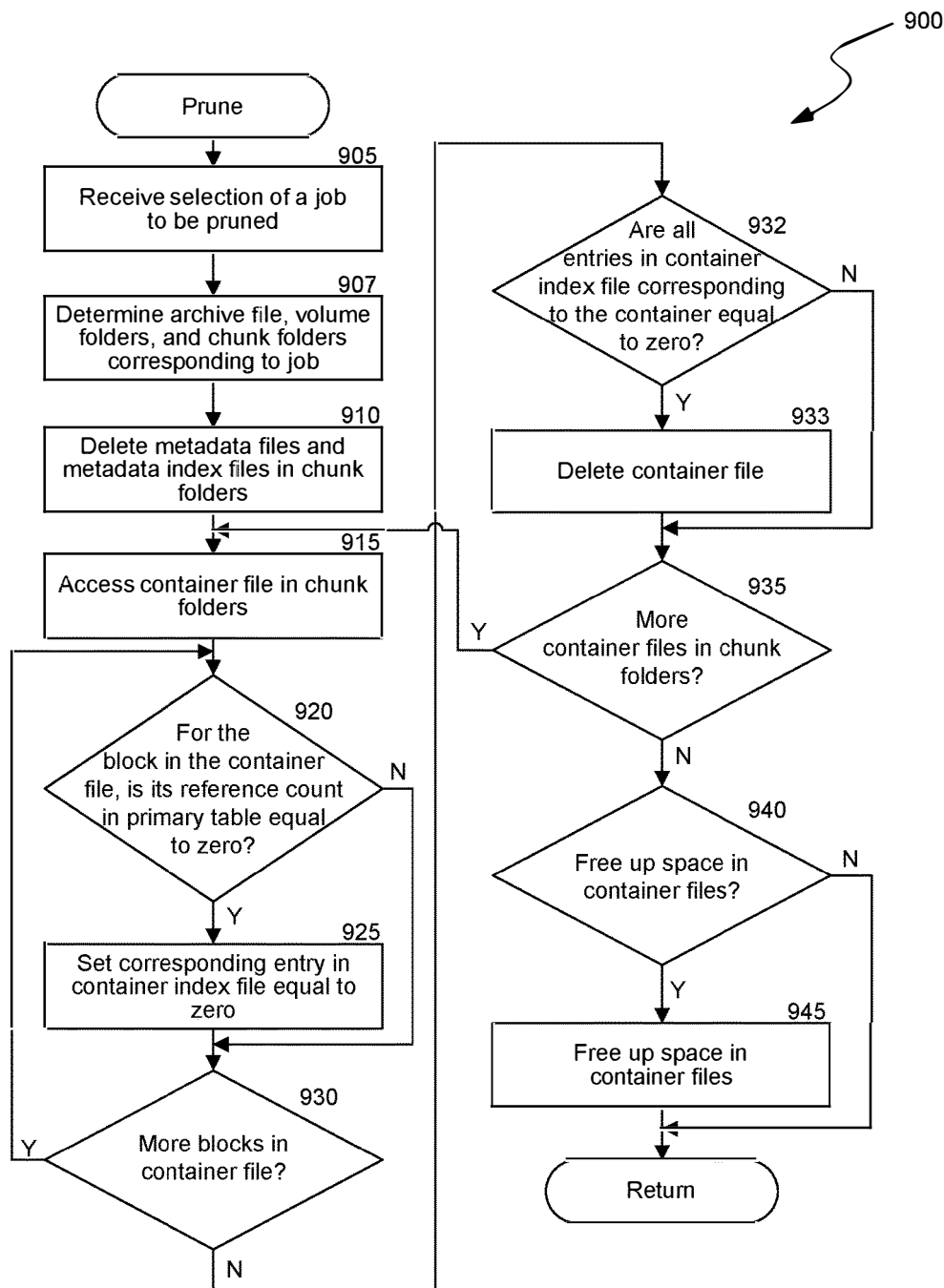


FIG. 9

U.S. Patent

Apr. 2, 2019

Sheet 13 of 33

US 10,248,657 B2

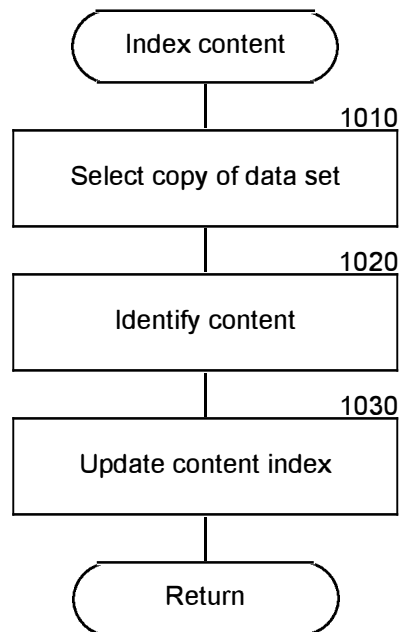


FIG. 10

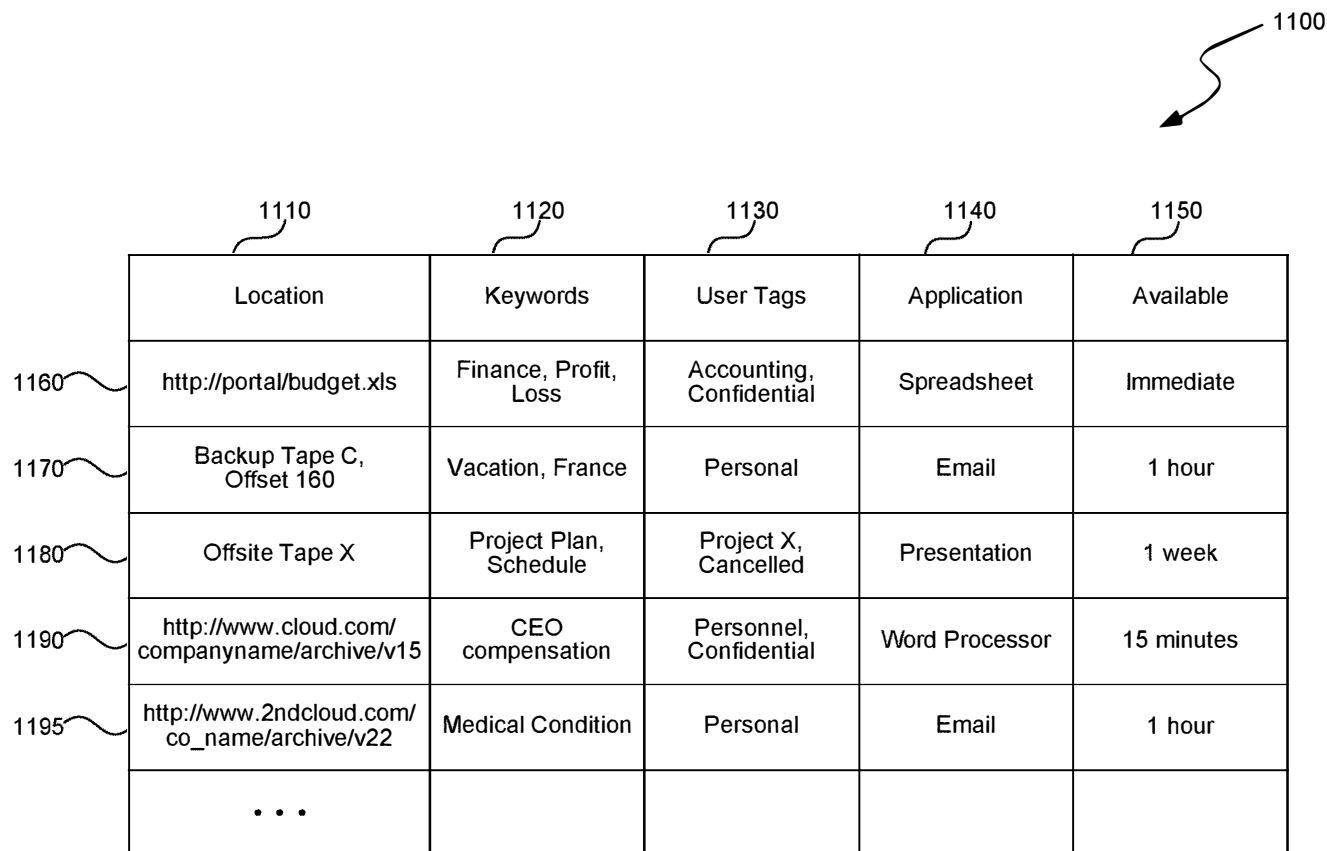


Diagram illustrating a table structure (FIG. 11) with columns and rows. The table is labeled 1100. The columns are labeled 1110, 1120, 1130, 1140, and 1150. The rows are labeled 1160, 1170, 1180, 1190, and 1195. The table contains the following data:

	1110 Location	1120 Keywords	1130 User Tags	1140 Application	1150 Available
1160	http://portal/budget.xls	Finance, Profit, Loss	Accounting, Confidential	Spreadsheet	Immediate
1170	Backup Tape C, Offset 160	Vacation, France	Personal	Email	1 hour
1180	Offsite Tape X	Project Plan, Schedule	Project X, Cancelled	Presentation	1 week
1190	http://www.cloud.com/companyname/archive/v15	CEO compensation	Personnel, Confidential	Word Processor	15 minutes
1195	http://www.2ndcloud.com/co_name/archive/v22	Medical Condition	Personal	Email	1 hour
	...				

FIG. 11

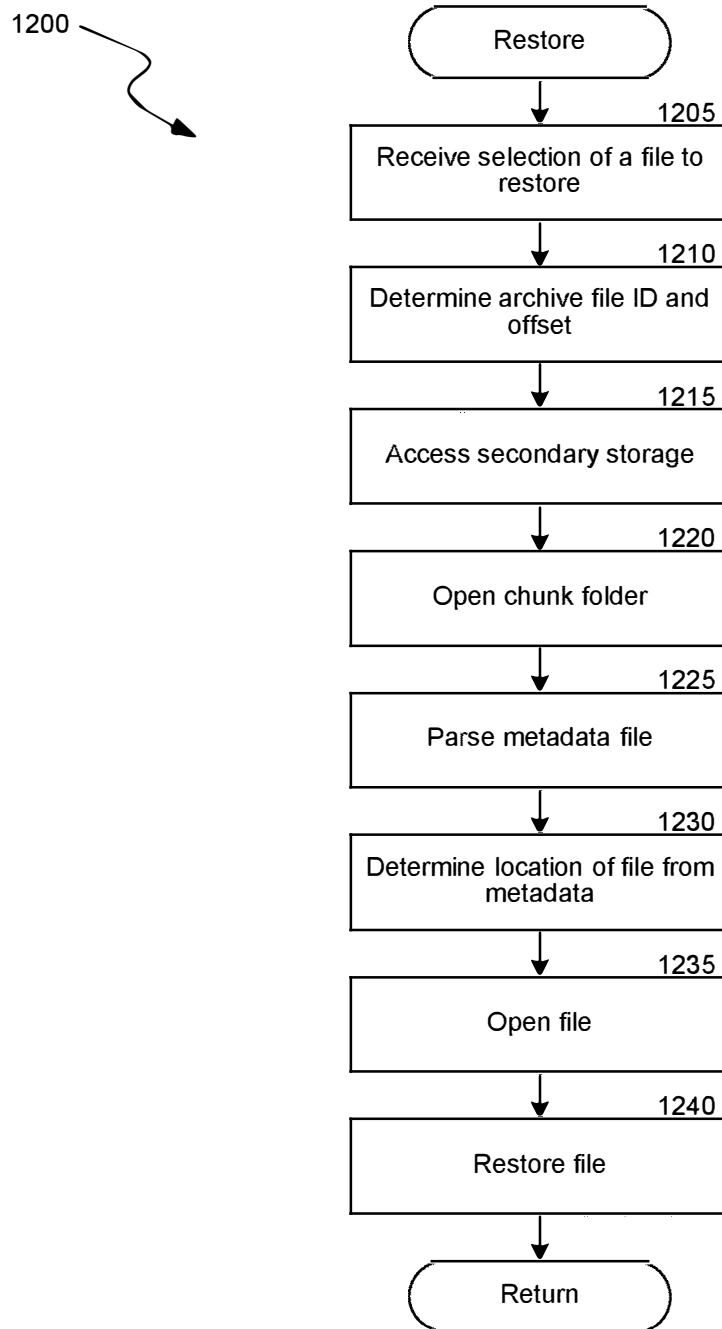


FIG. 12

1300

1310	1320	1330
Archive File ID	File ID	Offset
AF ₁	F ₁	OF ₁
	F ₂	OF ₂
	F ₃	OF ₃

	F _N	OF _n

FIG. 13A

1350

1370	1380	1390
Archive File ID	Media Chunk	Start
C, J, Cycle, AF	M ₁ , C ₁	AF ₁ , OF ₁ , Size
	M ₁ , C ₂	AF ₁ , OF ₂ , Size
	M ₂ , C ₃	AF ₁ , OF ₃ , Size

FIG. 13B

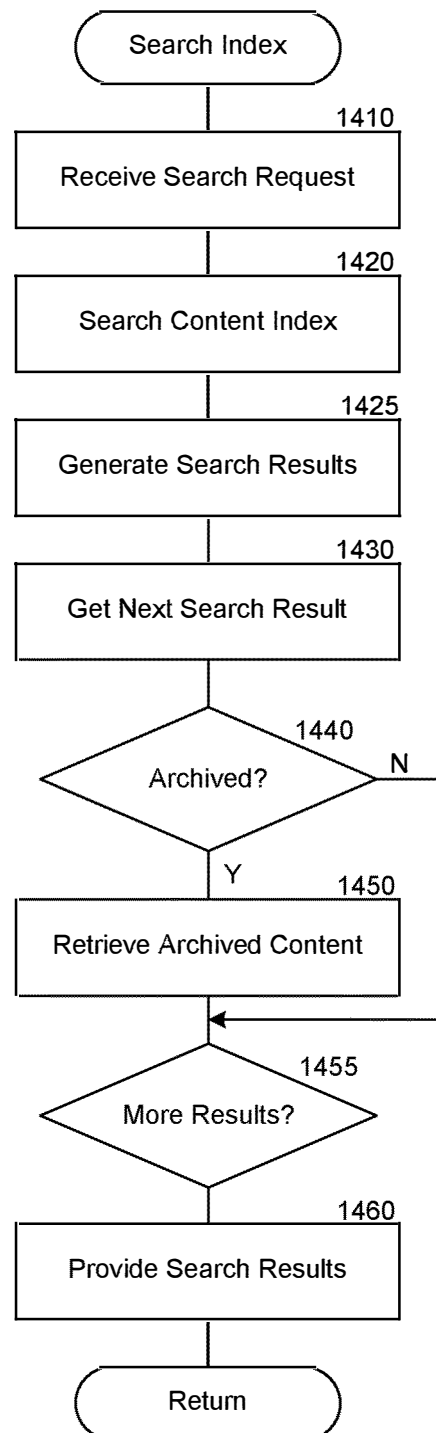


FIG. 14

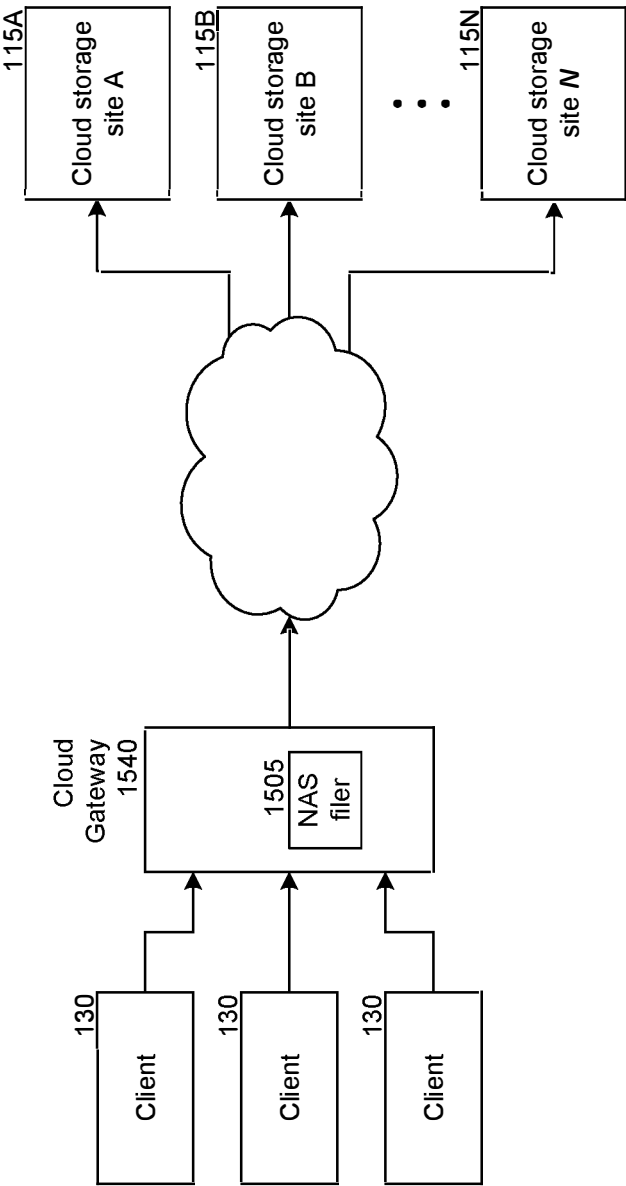


FIG. 15

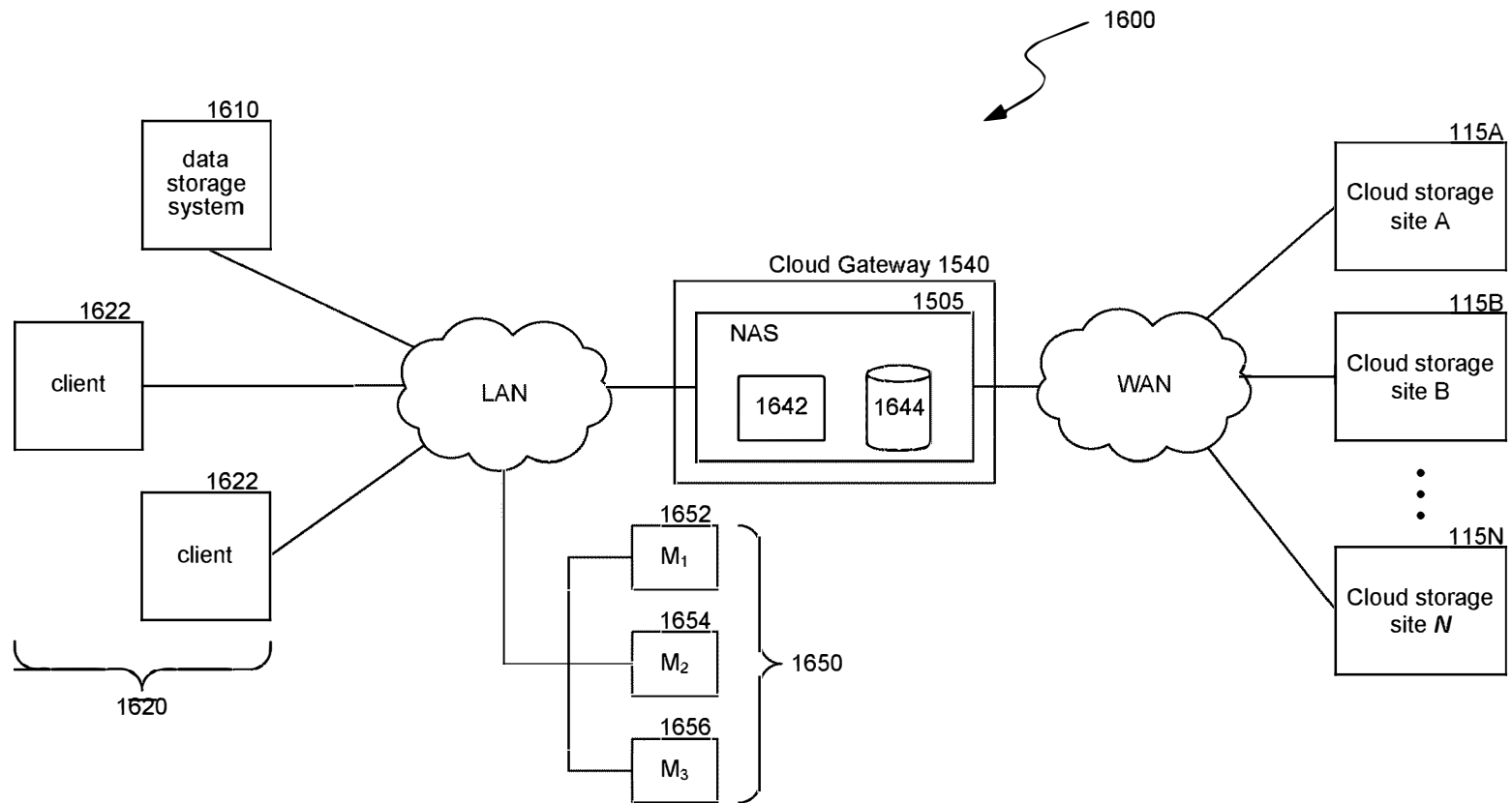


FIG. 16

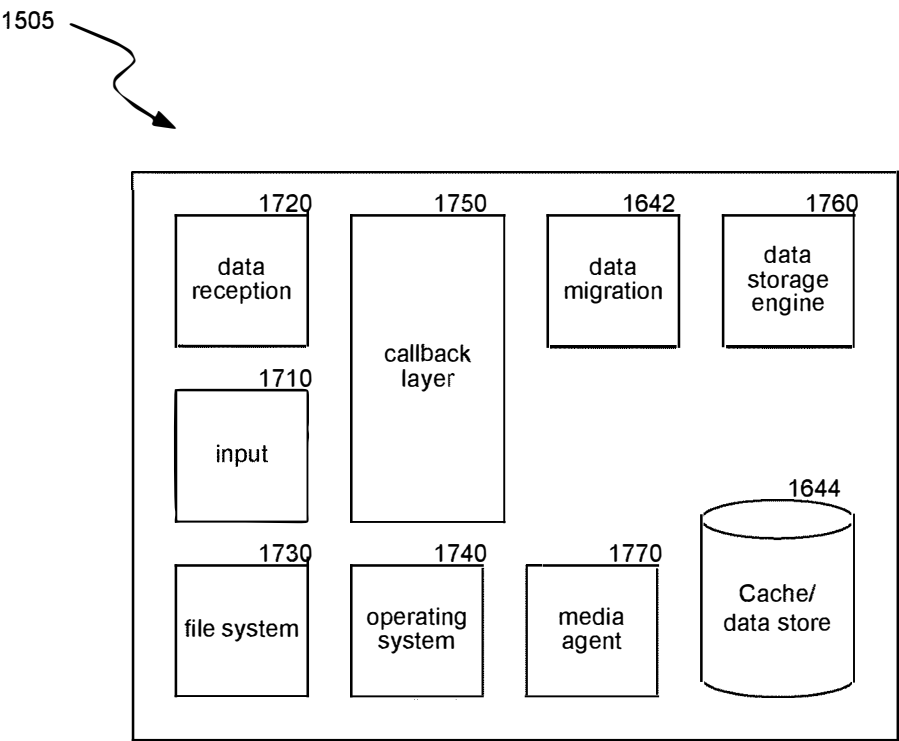


FIG. 17

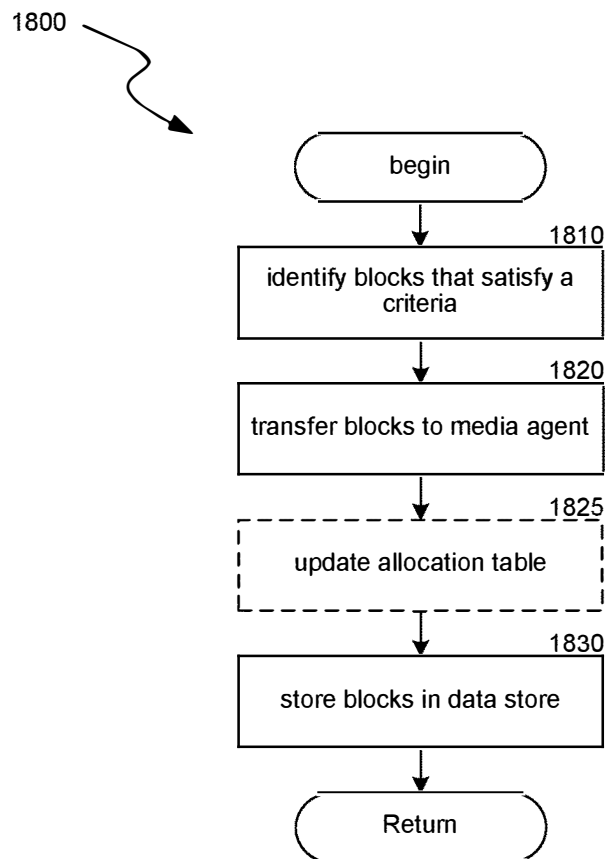


FIG. 18

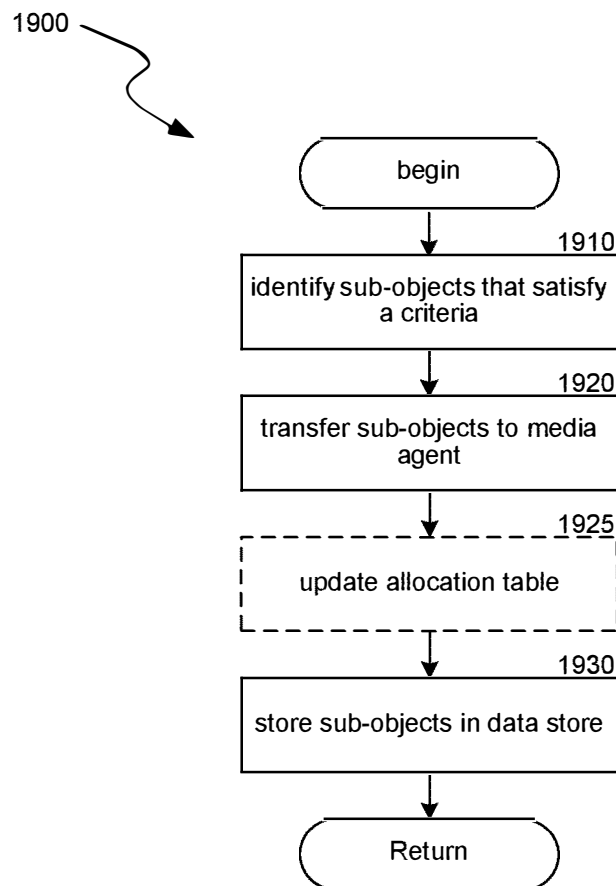


FIG. 19

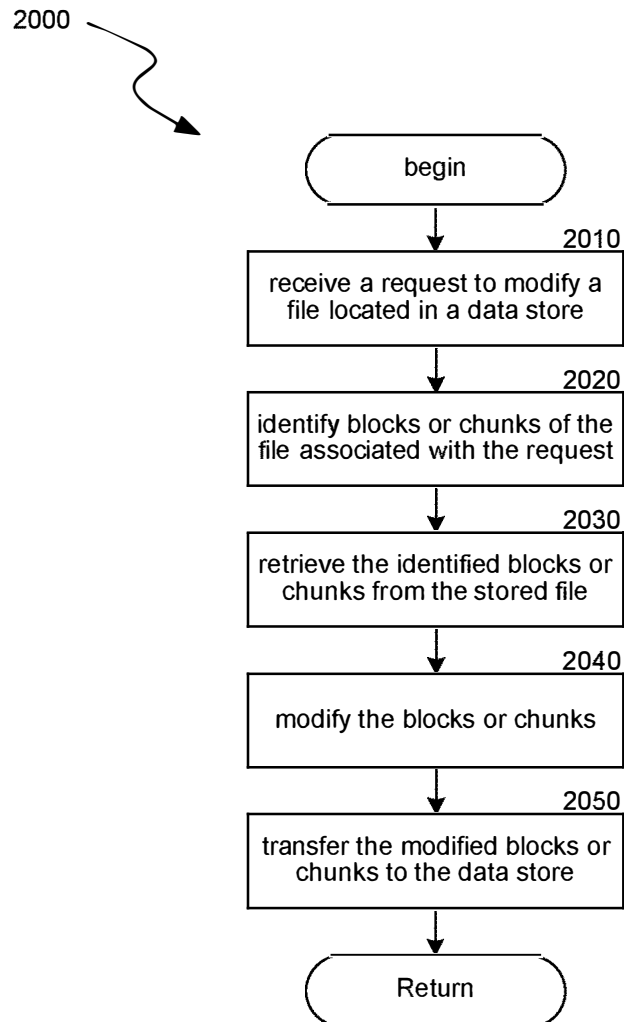


FIG. 20

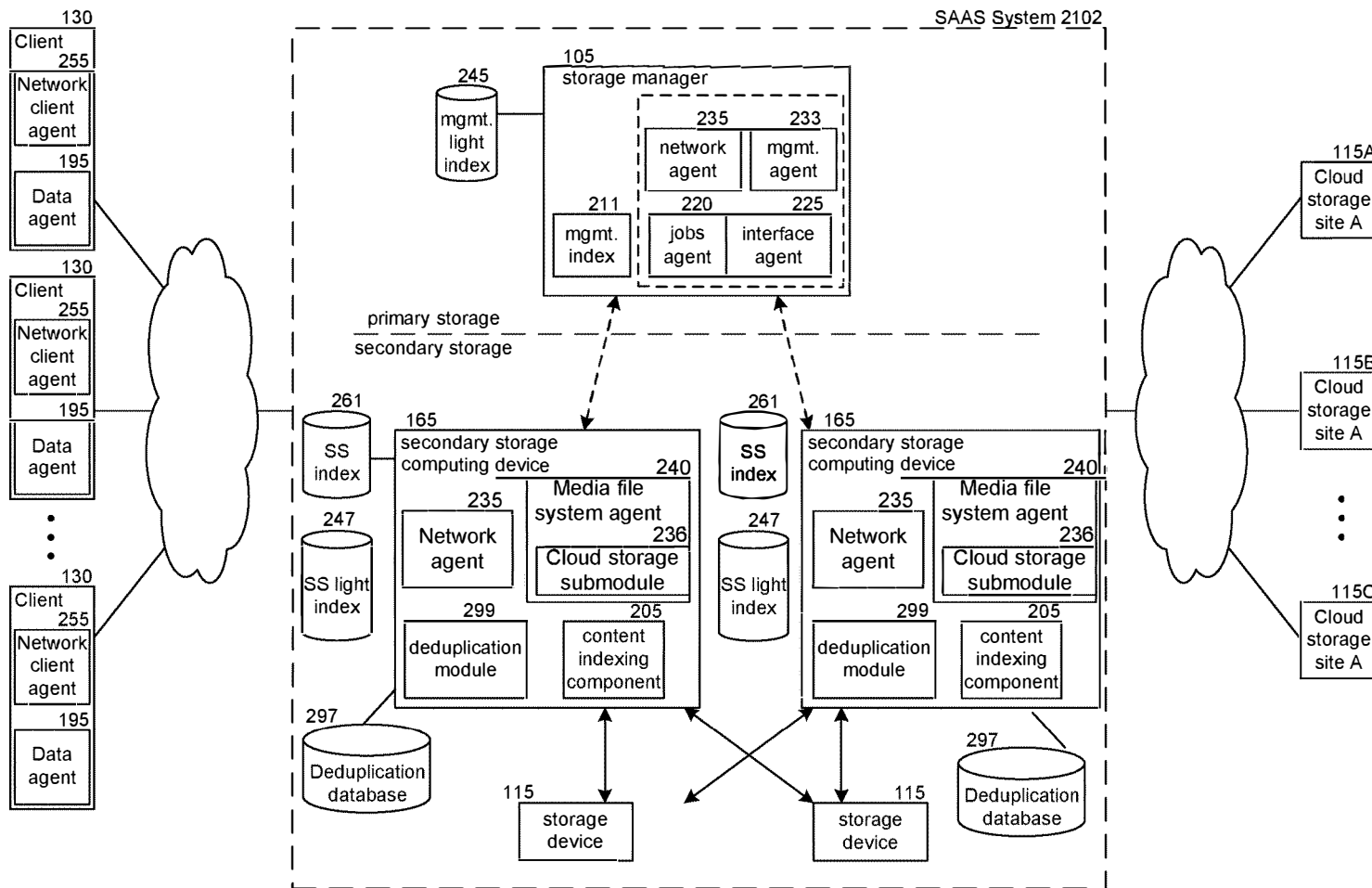


FIG. 21

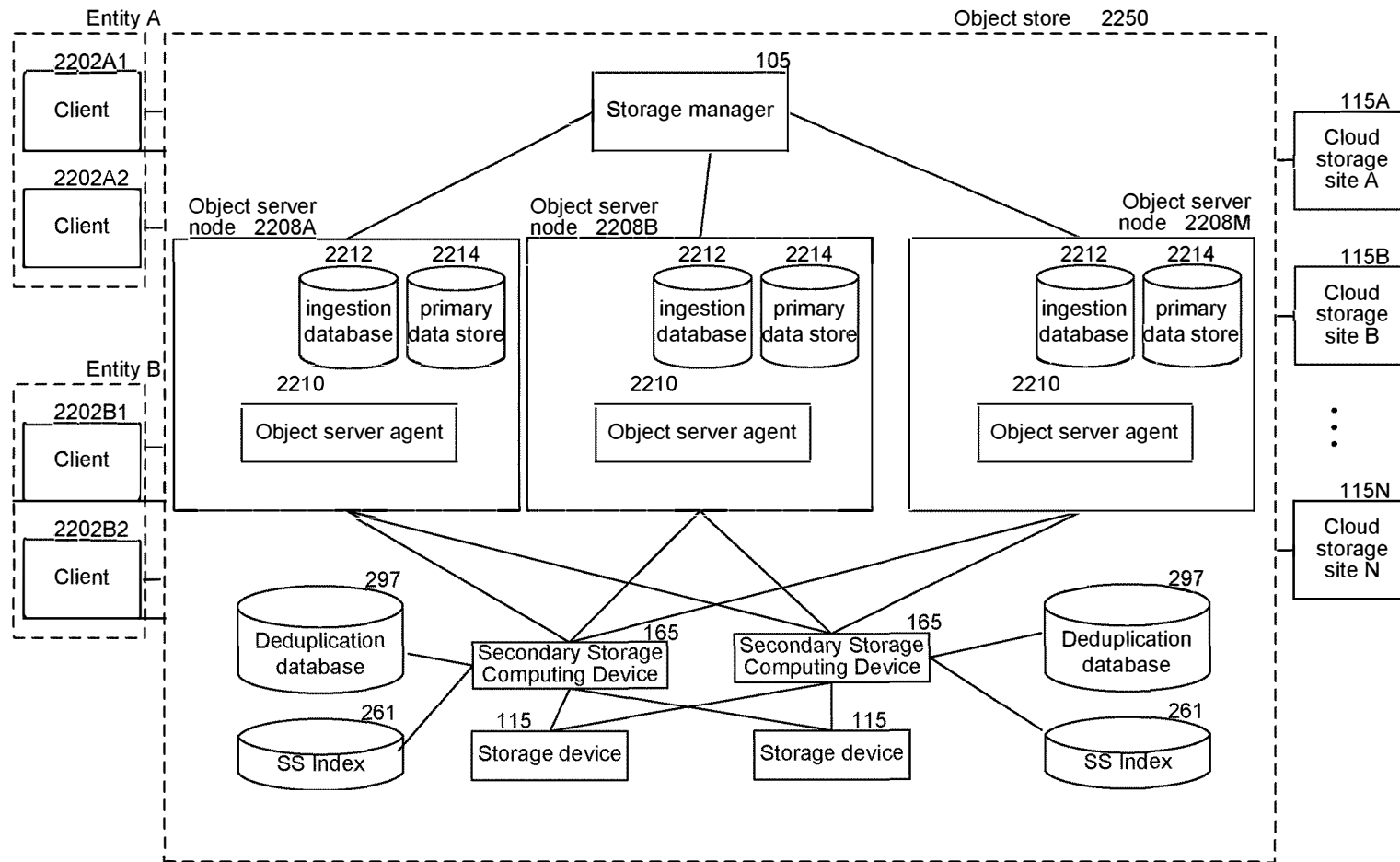
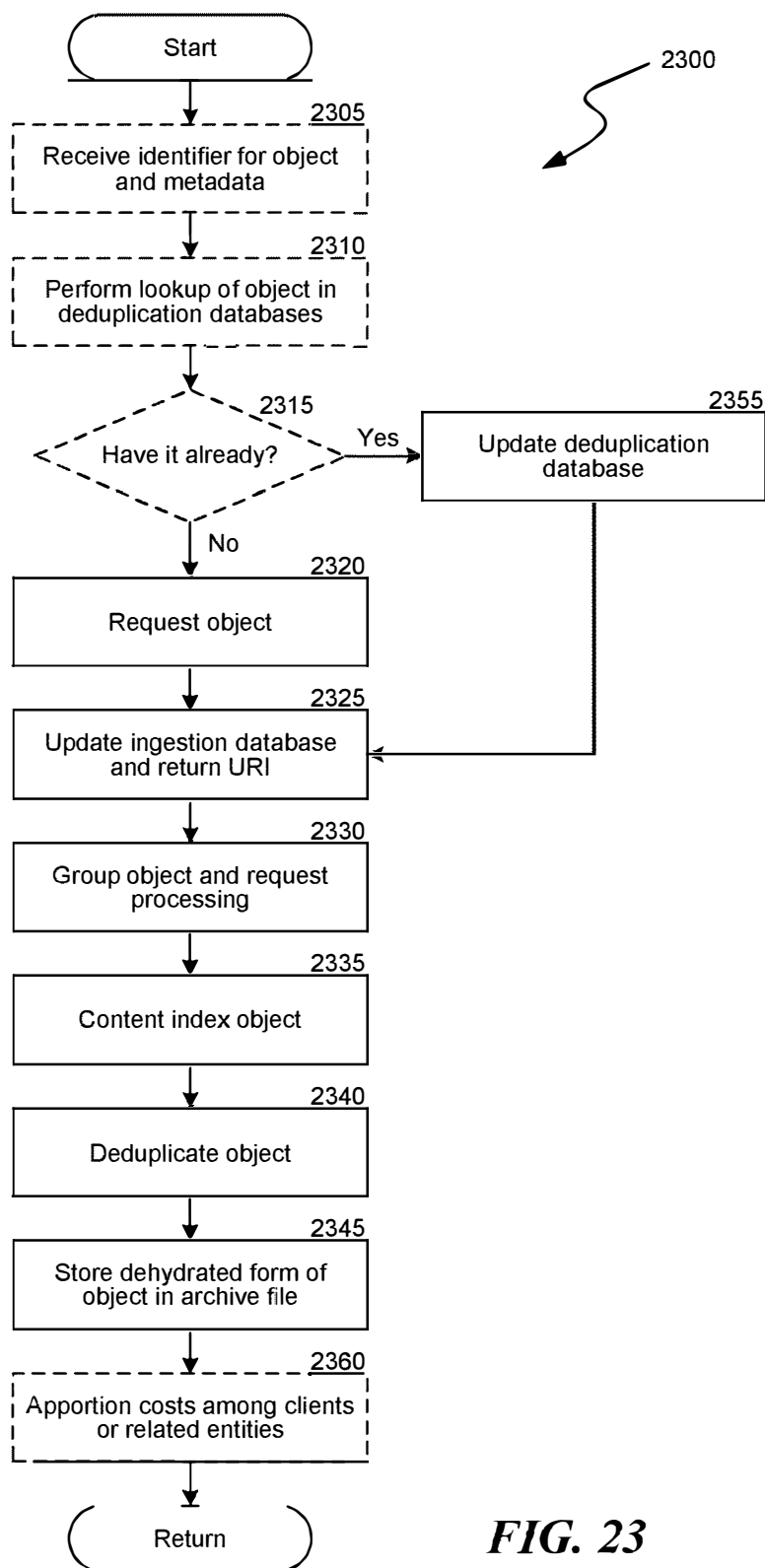


FIG. 22

U.S. Patent**Apr. 2, 2019****Sheet 26 of 33****US 10,248,657 B2****FIG. 23**

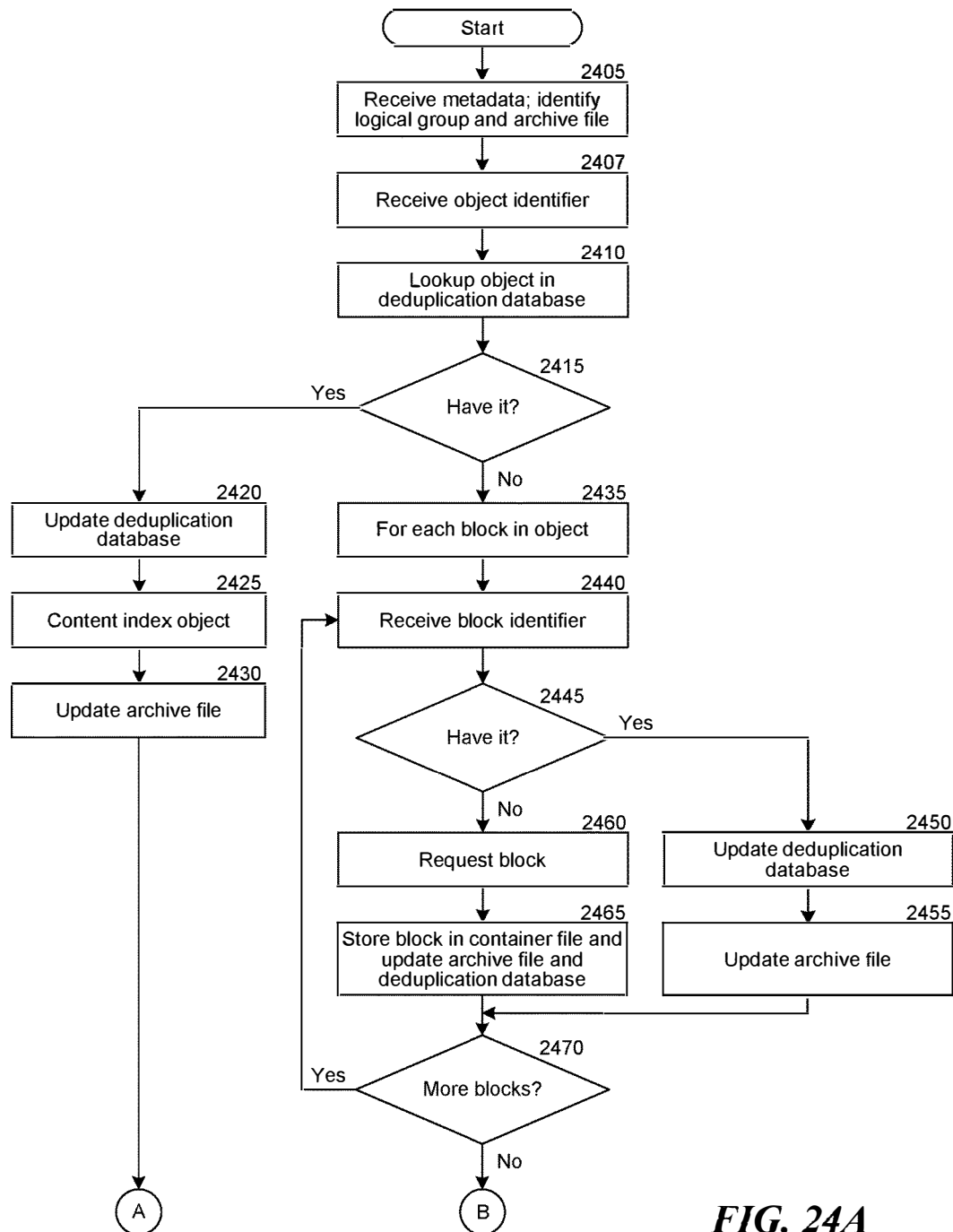


FIG. 24A

U.S. Patent

Apr. 2, 2019

Sheet 28 of 33

US 10,248,657 B2

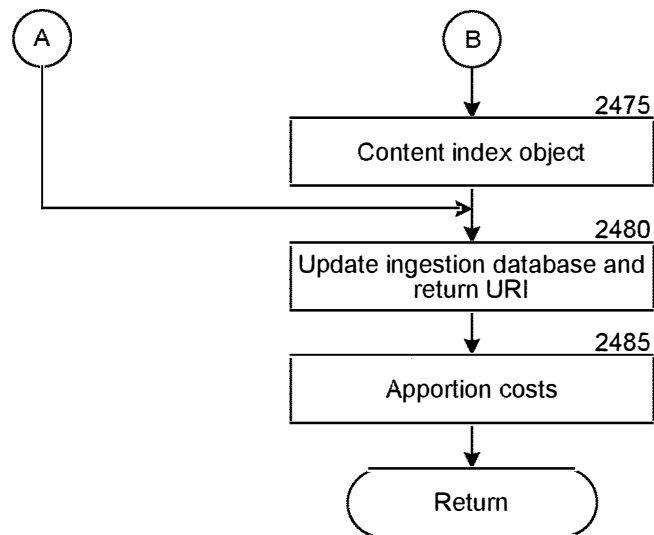


FIG. 24B

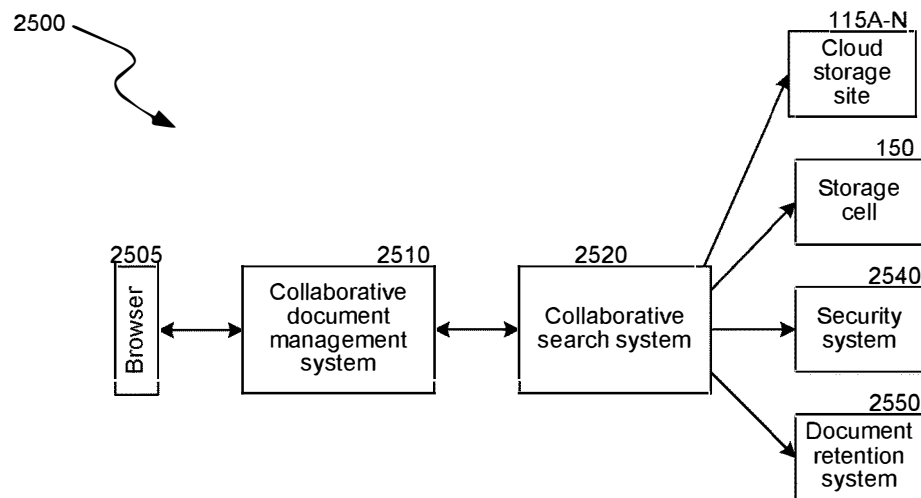


FIG. 25

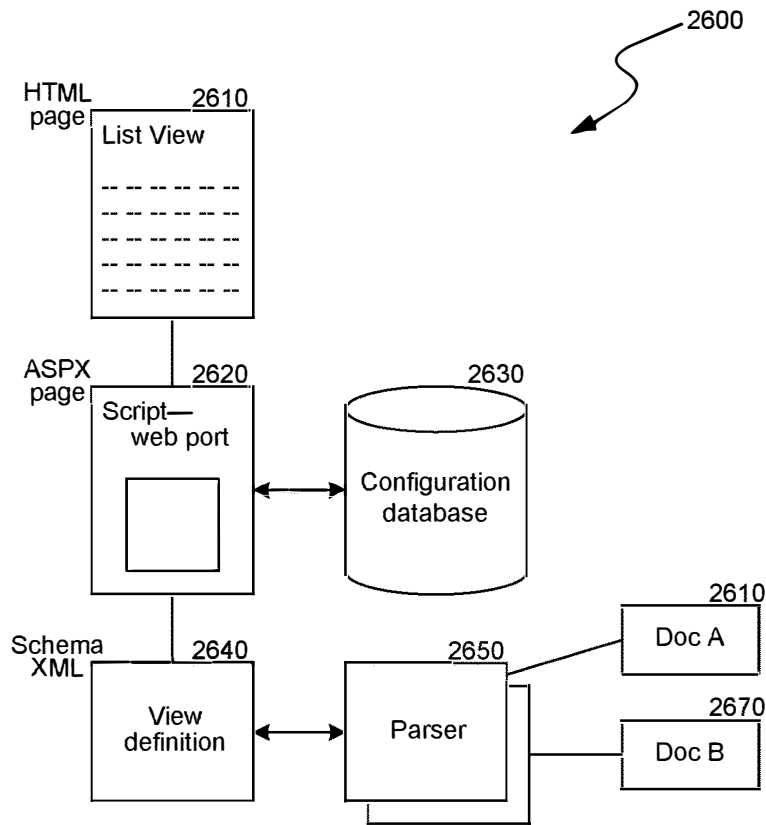
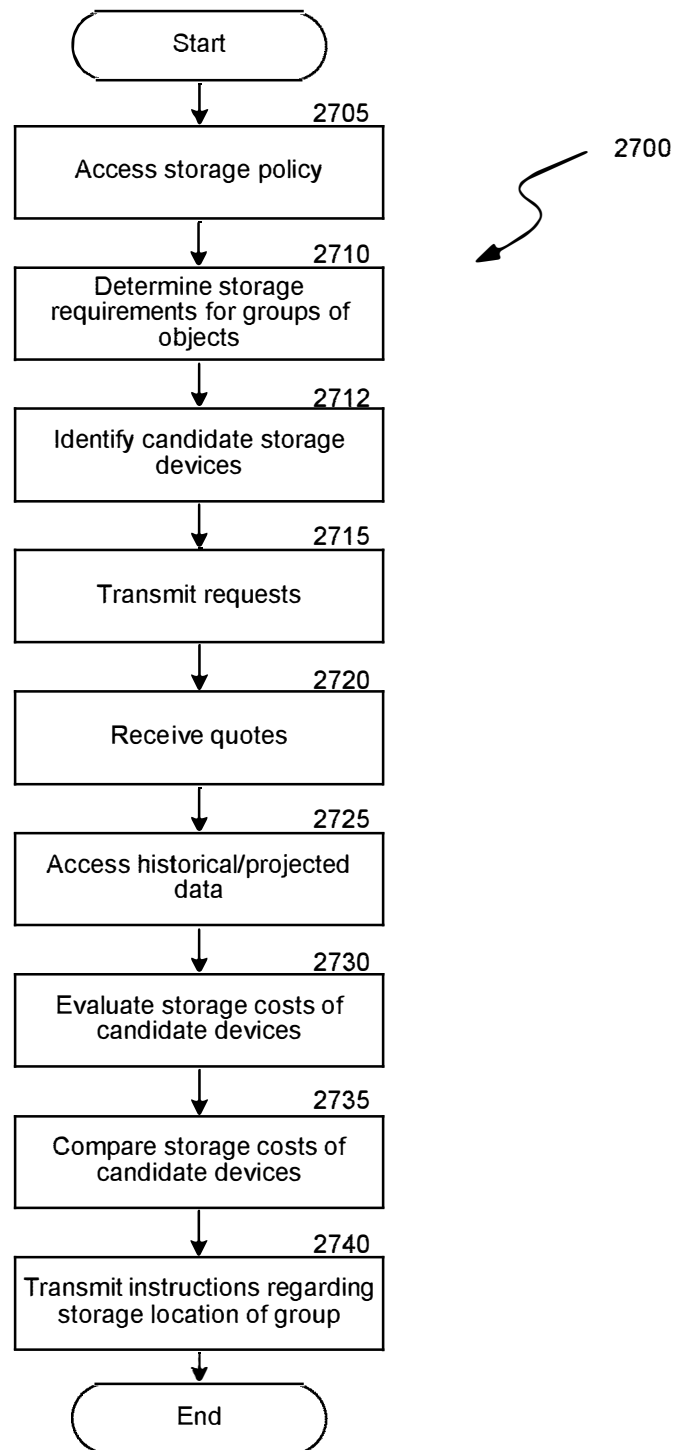


FIG. 26

**FIG. 27**

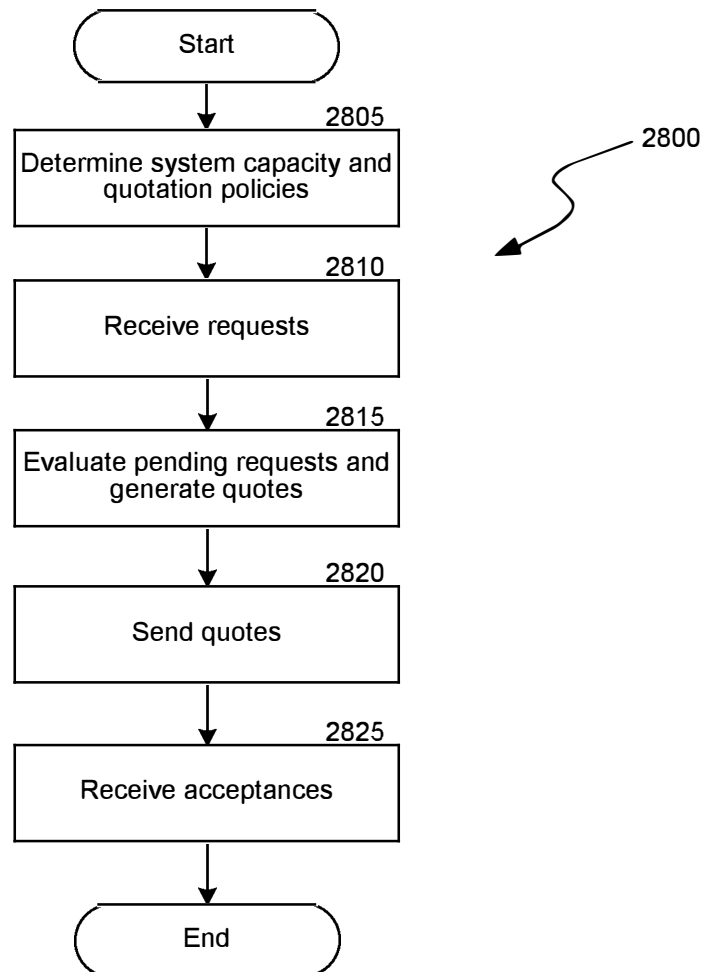
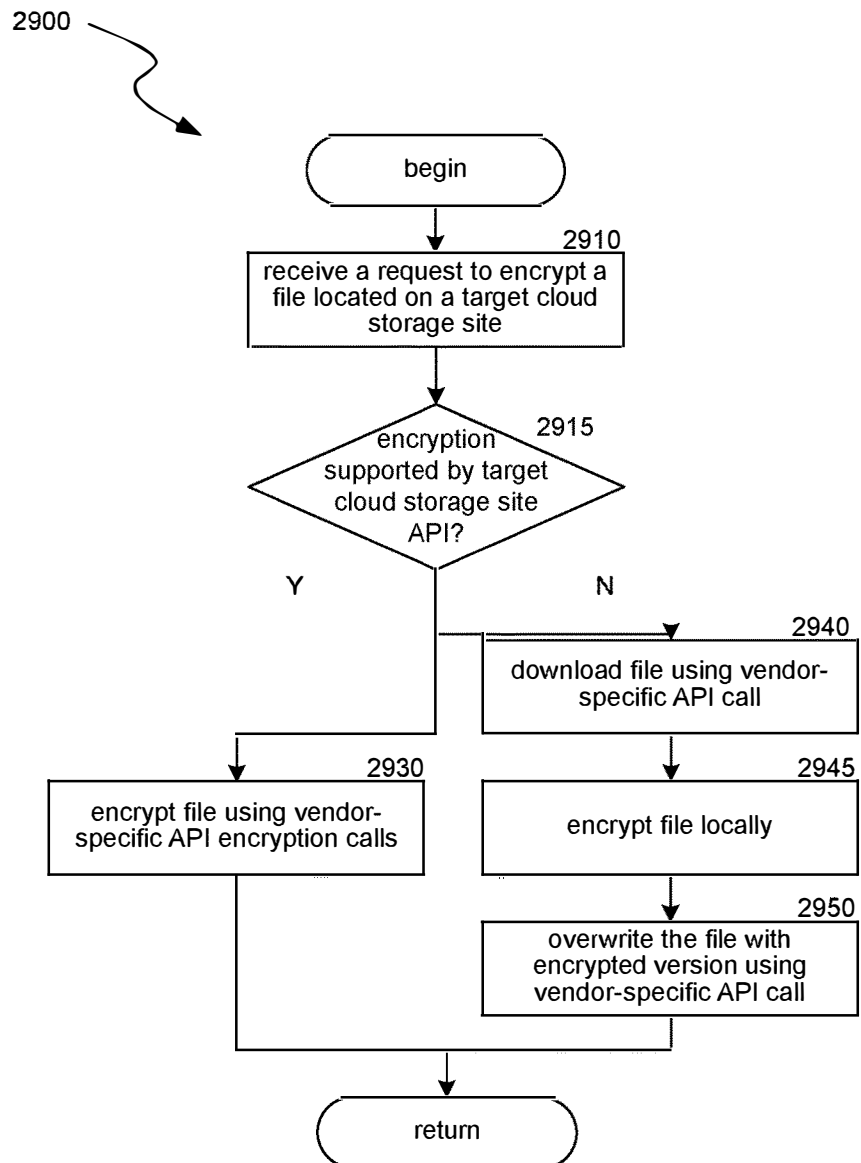


FIG. 28

**FIG. 29**

US 10,248,657 B2

1

**DATA OBJECT STORE AND SERVER FOR A
CLOUD STORAGE ENVIRONMENT,
INCLUDING DATA DEDUPLICATION AND
DATA MANAGEMENT ACROSS MULTIPLE
CLOUD STORAGE SITES**

**CROSS-REFERENCE TO RELATED
APPLICATIONS**

This application is a divisional of U.S. patent application Ser. No. 14/494,674, filed Sep. 24, 2014, now U.S. Pat. No. 9,454,537, entitled DATA OBJECT STORE AND SERVER FOR A CLOUD STORAGE ENVIRONMENT, INCLUDING DATA DEDUPLICATION AND DATA MANAGEMENT ACROSS MULTIPLE CLOUD STORAGE SITES, which is a continuation of U.S. patent application Ser. No. 13/615,999, filed Sep. 14, 2012, now U.S. Pat. No. 8,849,761, entitled DATA OBJECT STORE AND SERVER FOR A CLOUD STORAGE ENVIRONMENT, INCLUDING DATA DEDUPLICATION AND DATA MANAGEMENT ACROSS MULTIPLE CLOUD STORAGE SITES, which is a continuation of U.S. patent application Ser. No. 12/751,850, filed Mar. 31, 2010, now U.S. Pat. No. 8,285,681, entitled DATA OBJECT STORE AND SERVER FOR A CLOUD STORAGE ENVIRONMENT, INCLUDING DATA DEDUPLICATION AND DATA MANAGEMENT ACROSS MULTIPLE CLOUD STORAGE SITES, which claims the benefit of the assignee's U.S. Patent Application Nos. 61/299,313, filed Jan. 28, 2010, entitled PERFORMING DATA STORAGE OPERATIONS, INCLUDING CONTENT-INDEXING, CONTAINERIZED DEDUPLICATION, AND POLICY-DRIVEN STORAGE WITHIN A CLOUD ENVIRONMENT; 61/221,993, filed Jun. 30, 2009, entitled SYSTEMS AND METHODS FOR PERFORMING DATA STORAGE OPERATIONS, INCLUDING CROSS-CLOUD STORAGE, OVER VARIOUS NETWORK PROTOCOLS; and 61/223,695, filed Jul. 7, 2009, entitled SYSTEMS AND METHODS FOR PERFORMING DATA STORAGE OPERATIONS, INCLUDING CROSS-CLOUD STORAGE, OVER VARIOUS NETWORK PROTOCOLS, all of which are incorporated herein by reference.

BACKGROUND

Current storage management systems employ a number of different methods to perform storage operations on electronic data. For example, data can be stored in primary storage as a primary copy that includes production data, or in secondary storage as various types of secondary copies including, as a backup copy, a snapshot copy, a hierarchical storage management copy ("HSM"), as an archive copy, and as other types of copies.

A primary copy of data is generally a production copy or other "live" version of the data which is used by a software application and is generally in the native format of that application. Primary copy data may be maintained in a local memory or other high-speed storage device that allows for relatively fast data access if necessary. Such primary copy data is typically intended for short term retention (e.g., several hours or days) before some or all of the data is stored as one or more secondary copies, for example to prevent loss of data in the event a problem occurred with the data stored in primary storage.

Secondary copies include point-in-time data and are typically for intended for long-term retention (e.g., weeks, months or years depending on retention criteria, for example as specified in a storage policy as further described herein)

2

before some or all of the data is moved to other storage or discarded. Secondary copies may be indexed so users can browse, search and restore the data at another point in time. After certain primary copy data is backed up, a pointer or other location indicia such as a stub may be placed in the primary copy to indicate the current location of that data. Further details may be found in the assignee's U.S. Pat. No. 7,107,298, filed Sep. 30, 2002, entitled SYSTEM AND METHOD FOR ARCHIVING OBJECTS IN AN INFORMATION STORE (U.S. patent application Ser. No. 10/260,209).

One type of secondary copy is a backup copy. A backup copy is generally a point-in-time copy of the primary copy data stored in a backup format as opposed to in native application format. For example, a backup copy may be stored in a backup format that is optimized for compression and efficient long-term storage. Backup copies generally have relatively long retention periods and may be stored on media with slower retrieval times than other types of secondary copies and media. In some cases, backup copies may be stored at an offsite location.

Another form of secondary copy is a snapshot copy. From an end-user viewpoint, a snapshot may be thought of as an instant image of the primary copy data at a given point in time. A snapshot may capture the directory structure of a primary copy volume at a particular moment in time, and may also preserve file attributes and contents. In some embodiments, a snapshot may exist as a virtual file system, parallel to the actual file system. Users may gain a read-only access to the record of files and directories of the snapshot. By electing to restore primary copy data from a snapshot taken at a given point in time, users may also return the current file system to the prior state of the file system that existed when the snapshot was taken.

A snapshot may be created nearly instantly, using a minimum of file space, but may still function as a conventional file system backup. A snapshot may not actually create another physical copy of all the data, but may simply create pointers that are able to map files and directories to specific disk blocks.

In some embodiments, once a snapshot has been taken, subsequent changes to the file system typically do not overwrite the blocks in use at the time of snapshot. Therefore, the initial snapshot may use only a small amount of disk space to record a mapping or other data structure representing or otherwise tracking the blocks that correspond to the current state of the file system. Additional disk space is usually only required when files and directories are actually modified later. Furthermore, when files are modified, typically only the pointers which map to blocks are copied, not the blocks themselves. In some embodiments, for example in the case of copy-on-write snapshots, when a block changes in primary storage, the block is copied to secondary storage before the block is overwritten in primary storage and the snapshot mapping of file system data is updated to reflect the changed block(s) at that particular point in time.

An HSM copy is generally a copy of the primary copy data, but typically includes only a subset of the primary copy data that meets a certain criteria and is usually stored in a format other than the native application format. For example, an HSM copy might include only that data from the primary copy that is larger than a given size threshold or older than a given age threshold and that is stored in a backup format. Often, HSM data is removed from the primary copy, and a stub is stored in the primary copy to indicate its new location. When a user requests access to the

US 10,248,657 B2

3

HSM data that has been removed or migrated, systems use the stub to locate the data and often make recovery of the data appear transparent even though the HSM data may be stored at a location different from the remaining primary copy data.

An archive copy is generally similar to an HSM copy, however, the data satisfying criteria for removal from the primary copy is generally completely removed with no stub left in the primary copy to indicate the new location (i.e., where it has been moved to). Archive copies of data are generally stored in a backup format or other non-native application format. In addition, archive copies are generally retained for very long periods of time (e.g., years) and in some cases are never deleted. Such archive copies may be made and kept for extended periods in order to meet compliance regulations or for other permanent storage applications.

In some embodiments of storage management systems, application data over its lifetime moves from more expensive quick access storage to less expensive slower access storage. This process of moving data through these various tiers of storage is sometimes referred to as information lifecycle management ("ILM"). This is the process by which data is "aged" from more forms of secondary storage with faster access/restore times down through less expensive secondary storage with slower access/restore times, for example, as the data becomes less important or mission critical over time.

In some embodiments, storage management systems may perform additional operations upon copies, including deduplication, content indexing, data classification, data mining or searching, electronic discovery (E-discovery) management, collaborative searching, encryption and compression.

One example of a system that performs storage operations on electronic data that produce such copies is the Simpiana storage management system by CommVault Systems of Oceanport, N.J. The Simpiana system leverages a modular storage management architecture that may include, among other things, storage manager components, client or data agent components, and media agent components as further described in U.S. Pat. No. 7,246,207, filed Apr. 5, 2004, entitled SYSTEM AND METHOD FOR DYNAMICALLY PERFORMING STORAGE OPERATIONS IN A COMPUTER NETWORK. The Simpiana system also may be hierarchically configured into backup cells to store and retrieve backup copies of electronic data as further described in U.S. Pat. No. 7,395,282, filed Jul. 15, 1999, entitled HIERARCHICAL BACKUP AND RETRIEVAL SYSTEM.

Components within conventional storage management systems often communicate via one or more proprietary network protocols; this limits the devices that may connect to the system. Conventional systems may utilize propriety or non-proprietary network protocols at any of the seven Open Systems Interconnection Reference Model (OSIRM) layers, and may often utilize proprietary application-layer protocols. For example, if a client has primary data stored on it, and a storage management system is utilized to create a secondary copy of this data on a secondary storage device, the client may communicate with the secondary storage device by utilizing a proprietary application-level network protocol. In order to create a secondary copy on the secondary storage device in such a scenario, both the client and secondary storage device must have proprietary software and/or hardware installed or otherwise be configured to perform the proprietary network protocol. Thus, the ability of a conventional storage management system is generally

4

limited to performing storage operations on those clients and secondary storage devices having pre-installed hardware or software.

Although some conventional data storage systems may permit a client to communicate with the system via a non-proprietary network protocol such as hypertext transfer protocol (HTTP) or file transfer protocol (FTP), generally such systems do not facilitate a wide range of value-added storage operations. For example, cloud storage sites typically provide only storage of and access to data objects as a service provided to end users. Generally, uploading, access and manipulation of data stored on a cloud storage site is conducted via an HTTP, FTP or similar network connection. Cloud storage service providers include Amazon Simple Storage Service, Rackspace, Windows Azure, and Iron Mountain, and Nirvanix Storage Delivery Network. Cloud storage service providers often bill end users on a utility computing basis, e.g., per gigabyte stored, uploaded and/or downloaded per month. Conventional cloud storage sites may not permit the end user to perform value-added storage operations such as ILM, deduplication, content indexing, data classification, data mining or searching, E-discovery management, collaborative searching, encryption or compression.

The need exists for systems and methods that overcome the above problems, as well as systems and methods that provide additional benefits. Overall, the examples herein of some prior or related systems and methods and their associated limitations are intended to be illustrative and not exclusive. Other limitations of existing or prior systems and methods will become apparent to those of skill in the art upon reading the following Detailed Description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates an example of one arrangement of resources in a computing network that may employ aspects of the invention.

FIG. 2 is a block diagram illustrating an example of a data storage enterprise system that may employ aspects of the invention.

FIG. 3A is a flow diagram illustrating a routine for writing data to cloud storage sites.

FIG. 3B, is a flow diagram illustrating a routine for migrating or copying data into an archive format in secondary storage, including secondary cloud storage.

FIG. 4 is a block diagram illustrating an example of a deduplication module.

FIGS. 5A-5D illustrate various data structures for deduplicating and storing copies or instances of data objects on a storage device or for other processes.

FIG. 6 is a flow diagram illustrating a process for pruning a deduplication database by pruning or deleting data objects stored in archive files, or entire archive files.

FIGS. 7A-7C illustrate various data structures which aspects of the invention may utilize for pruning object-level deduplicated data or for other processes.

FIG. 8 illustrates various data structures which aspects of the invention may utilize for deduplicating and storing copies or instances of data blocks on a storage device or for other processes.

FIG. 9 is a flow diagram illustrating a process for pruning a deduplication database by pruning or deleting data blocks stored in archive files, or entire archive files.

FIG. 10 is a flow diagram that illustrates the processing of a content indexing component.

US 10,248,657 B2

5

FIG. 11 illustrates suitable data structures for facilitating content indexing.

FIG. 12 is a flow diagram illustrating a process for restoring or retrieving data from chunk folders in an archive file format on secondary storage.

FIGS. 13A and 13B illustrate example data structures that the system may maintain to facilitate the restoration or retrieval of data from chunk folders in an archive file format on secondary storage.

FIG. 14 is a flow diagram illustrating the processing of a search request by the system.

FIG. 15 illustrates another example of an arrangement of resources in a computing network that may employ aspects of the invention.

FIG. 16 is a block diagram illustrating a suitable environment for utilizing a networked data storage device.

FIG. 17 shows a block diagram illustrating components of the network-attached storage (NAS) filer component of a cloud gateway configured to perform data migration.

FIG. 18 depicts a flow diagram illustrating a routine for performing block-level data migration in a cloud gateway.

FIG. 19 is a flow diagram illustrating a routine for performing sub-object-level data migration in a cloud gateway.

FIG. 20 shows a flow diagram illustrating a routine for block-based or sub-object-based data restoration and modification in a cloud gateway.

FIG. 21 illustrates another example of an arrangement of resources in a computing network that may employ aspects of the invention to provide data storage software as a service.

FIG. 22 is a block diagram illustrating components of an object store.

FIG. 23 shows a flow diagram illustrating a first process that may be performed by an object store to process a request to store a data object.

FIGS. 24A and 24B together show a flow diagram illustrating a second process that may be performed by an object store to process a request to store a data object.

FIG. 25 is a block diagram illustrating an example architecture for integrating a collaborative search system with a collaborative document management system.

FIG. 26 is a schematic diagram illustrating integration of parsers with a typical collaborative document management system.

FIG. 27 is a flow diagram of a process for identifying suitable storage locations for various data objects subject to a storage policy.

FIG. 28 is a flow diagram of a process for scheduling cloud storage requests.

FIG. 29 illustrates a process for encrypting files stored within a cloud storage site.

DETAILED DESCRIPTION

The headings provided herein are for convenience only and do not necessarily affect the scope or meaning of the claimed invention.

1. Overview

With the massive volume of files being hosted in cloud environments, traditional file system based approaches are failing to scale. As much as 90% of new data created is unstructured and/or file based. As such data makes its way into the cloud, the need for systems that can scale to several million files and possibly petabytes of capacity becomes necessary. Traditional file systems and filers have their strengths, and high-performance file sharing needs still exist

6

within data centers, so existing filers and file systems fulfill that need. Cloud storage, on the other hand, with associated network latencies is not always a good fit for certain use cases. But cloud storage excels with Internet applications where the generation of content can be viral and where it can be virtually impossible to predict capacity or access needs. Cloud storage is also ideal in the case of Web 2.0 applications which promote collaboration between hundreds and thousands of user sharing the same files or objects.

While file systems have been a successful way of allowing people to store their data in an intuitive form that is easy to visualize, they have complexities which get exposed when the number of objects they need to manage reach massive proportions. File systems are typically built on block storage devices and all files are eventually broken down into blocks that need to be placed on the storage system. The file system has to maintain a "table of contents" (e.g. a FAT), which tracks not only what files it is holding, but which blocks on the storage comprise that file. On a system with a massive number of files, each with a large number of blocks, the numbers get large enough that traditional file systems start to slow down or even crash. What's typically done when this happens is that a new file system or filer is added. But the new file system provides a completely different namespace than the original and all users of the file system (humans and applications) need to be aware of this change and know which namespace they need to look in to find their files.

Systems and methods are disclosed herein for performing data storage operations, including content indexing, containerized deduplication, and policy-driven storage, within a cloud environment. The systems support a variety of clients and storage devices that connect to the system in a cloud environment, which permits data transfer over wide area networks, such as the Internet, and which may have appreciable latency and/or packet loss. The system allows available storage devices to include cloud storage sites. Methods are disclosed for content indexing data stored within a cloud environment to facilitate later searching, including collaborative searching. Methods are also disclosed for performing containerized deduplication to reduce the strain on a system namespace and effectuate cost savings. Methods are disclosed for identifying suitable storage locations, including suitable cloud storage sites, for data files subject to a storage policy. Further, systems and methods for providing a cloud gateway and a scalable data object store within a cloud environment are disclosed.

Various examples of the invention will now be described. The following description provides specific details for a thorough understanding and enabling description of these examples. One skilled in the relevant art will understand, however, that the invention may be practiced without many of these details. Likewise, one skilled in the relevant art will also understand that the invention may include many other obvious features not described in detail herein. Additionally, some well-known structures or functions may not be shown or described in detail below, so as to avoid unnecessarily obscuring the relevant description.

The terminology used below is to be interpreted in its broadest reasonable manner, even though it is being used in conjunction with a detailed description of certain specific examples of the invention. Indeed, certain terms may even be emphasized below; however, any terminology intended to be interpreted in any restricted manner will be overtly and specifically defined as such in this Detailed Description section.

US 10,248,657 B2

7

Unless described otherwise below, aspects of the invention may be practiced with conventional data processing and data storage systems. Thus, the construction and operation of the various blocks shown in the Figures may be of conventional design, and need not be described in further detail herein to make and use aspects of the invention, because such blocks will be understood by those skilled in the relevant art. One skilled in the relevant art can readily make any modifications necessary to the blocks in the Figures based on the detailed description provided herein.

2. Suitable Environments

The Figures and the discussion herein provide a brief, general description of certain suitable computing environments in which aspects of the invention can be implemented. Although not required, aspects of the invention are described in the general context of computer-executable instructions, such as routines executed by a general-purpose computer, e.g., a server computer, wireless device, or personal computer. Those skilled in the relevant art will appreciate that aspects of the invention can be practiced with other communications, data processing, or computer system configurations, including: Internet appliances, hand-held devices (including personal digital assistants (PDAs), wearable computers, all manner of cellular or mobile phones, multi-processor systems, microprocessor-based or programmable consumer electronics, set-top boxes, network PCs, mini-computers, mainframe computers, and the like. The terms “computer,” “server,” and the like are generally used interchangeably herein, and refer to any of the above devices and systems, as well as any data processor. Aspects of the invention can be practiced in software that controls or operates data storage hardware that is specifically designed for use in data storage networks, e.g., as described in detail herein.

While aspects of the invention, such as certain functions, are described as being performed exclusively on a single device, the invention can also be practiced in distributed environments where functions or modules are shared among disparate processing devices, which are linked through a communications network, such as a Local Area Network (LAN), Wide Area Network (WAN), and/or the Internet. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Aspects of the invention including computer implemented instructions, data structures, screen displays, and other data may be stored or distributed on tangible computer-readable storage media, including magnetically or optically readable computer discs, hard-wired or preprogrammed chips (e.g., EEPROM semiconductor chips), nanotechnology memory, biological memory, or other data storage media. Alternatively, computer implemented instructions, data structures, screen displays, and other data under aspects of the invention may be distributed via communication medium, such as over the Internet or over other networks (including wireless networks), on a propagated signal on a propagation medium (e.g., an electromagnetic wave(s), a sound wave, etc.) over a period of time, or they may be provided on any analog or digital network (packet switched, circuit switched, or other scheme).

FIG. 1 illustrates an example of one arrangement of resources in a computing network that may employ the processes and techniques described herein, although many others are of course possible. Clients 130, as part of their function, may utilize data, which includes files, directories, metadata (e.g., access control list (ACLs) creation/edit dates associated with the data, etc.), and other data objects. The data on the clients 130 is typically a primary copy (e.g., a

8

production copy). During a copy, backup, archive or other storage operation, the clients 130 may send a copy of some data objects (or some components thereof) to a secondary storage computing device 165 by utilizing one or more data agents 195, described below.

The secondary storage computing device 165 may in turn create secondary copies of primary data objects (or some components thereof) in storage devices 115, which may include various cloud storage sites 115A-N. Communications between the secondary storage computing devices 165 and cloud storage sites 115A-N may utilize REST protocols (Representational state transfer interfaces) that satisfy basic C/R/U/D semantics (Create/Read/Update/Delete semantics), or other hypertext transfer protocol (“HTTP”)-based or file-transfer protocol (“FTP”)-based protocols (e.g. Simple Object Access Protocol).

In conjunction with creating secondary copies in cloud storage sites 115A-N, the secondary storage computing device 165 may also perform local content indexing and/or local object-level, sub-object-level or block-level deduplication when performing storage operations involving various cloud storage sites 115A-N. By providing content indexing and local searching, the system may reduce the time and cost associated with data access or data search requests sent to remote cloud storage sites. By deduplicating locally, the system may reduce the amount of data transfer required over a wide area network between the secondary storage computing devices 165 and the cloud storage sites 115A-N, and may reduce the cost associated with data uploads to and data storage on cloud storage sites. Further details are provided below.

3. Storage Operation Cell

FIG. 2 illustrates an example of one arrangement of a storage operation cell 150 in a computing network that may employ the processes and techniques described herein, although many others are of course possible. FIG. 2 shows a hierarchical arrangement of resources, which includes a storage operation cell 150 having a storage manager 105, one or more data agents 195, one or more network client agents 255, one or more secondary storage computing devices 165, one or more media file system agents 240, one or more storage devices 115, one or more clients 130, and one or more data or information stores 260. The cell 150 also includes a management index 211, a management light index 245, a jobs agent 220, an interface agent 225, a management agent 233, one or more network agents 235, one or more metabases 270, one or more secondary storage indices 261, one or more deduplication modules 299, one or more content indexing components 205, one or more deduplication databases 297, and one or more secondary storage light indices 247. Such system and elements represent a modular storage system such as the CommVault Simpana system, available from CommVault Systems, Inc. of Oceanport, N.J., and further described in the assignee’s U.S. Pat. No. 7,035,880, filed Jul. 6, 2000, entitled MODULAR BACKUP AND RETRIEVAL SYSTEM USED IN CONJUNCTION WITH A STORAGE AREA NETWORK. Although not illustrated in FIG. 1, in some implementations, one or more of the secondary storage computing devices 165 (and/or deduplication databases, secondary storage indices, secondary storage light indices, and/or other system components) may reside on one or more cloud storage site 115A-N. For example, in such implementations, a secondary storage computing device may utilize computational resources (e.g., computational processing capacity) provided by a vendor that operates a cloud storage site 115A-N to perform its functionality.

US 10,248,657 B2

9

A storage operation cell, such as cell **150**, may generally include combinations of hardware and software components associated with performing storage operations on electronic data. (While aspects of the invention are described as employing the hierarchical architecture with cells, those aspects may likewise be employed in other architectures without cells, such as a simple client-server or peer-to-peer configuration.) Storage operation cells **150** may be related to backup cells and provide some or all of the functionality of backup cells as described in the assignee's U.S. Pat. No. 7,395,282 filed Jul. 15, 1999, entitled HIERARCHICAL BACKUP AND RETRIEVAL SYSTEM. However, storage operation cells may also perform additional types of storage operations and other types of storage management functions that are not generally offered by backup cells.

Additional data storage operations performed by storage operation cells **150** may include creating, storing, retrieving, and migrating primary storage data (e.g., data store **260**) and secondary storage data (which may include, for example, snapshot copies, backup copies, Hierarchical Storage Management (HSM) copies, archive copies, and other types of copies of electronic data) stored on storage devices **115**. In some embodiments, storage operation cells may perform additional storage operations upon copies, including ILM, deduplication, content indexing, data classification, data mining or searching, electronic discovery (E-discovery) management, collaborative searching, encryption and compression. Alternatively or additionally, a storage operation cell may make or retain disaster recovery copies, often as secondary, high-availability disk copies. Such cell may make secondary disk copies to disaster recovery (DR) locations using auxiliary copy or replication technologies. Storage operation cells **150** may also provide one or more integrated management consoles for users or system processes to interface with in order to perform certain storage operations on electronic data. Such integrated management consoles may be displayed at a central control facility or several similar consoles may be distributed throughout multiple network locations to provide global or geographically specific network data storage information.

In one example, storage operations may be performed according to various storage preferences, for example, as expressed by a user preference or a storage policy. A "storage policy" is generally a data structure or other information source that includes a set of preferences and other storage criteria associated with performing a storage operation. The preferences and storage criteria may include, but are not limited to, a storage location (or a class or quality of storage location), deduplication requirements, relationships between system components, network pathways to utilize in a storage operation, retention policies, data characteristics, compression or encryption requirements, preferred system components to utilize in a storage operation, the estimated or historic usage or cost associated with operating system components, frequency or use/access/etc. various time-related factors, single-instancing and/or deduplication information, and other criteria relating to a data storage or management operation. For example, a storage policy may indicate that certain data is to be stored in the storage device **115**, retained for a specified period of time before being aged to another tier of secondary storage, copied to the storage device **115** using a specified number of data streams, etc. As one example, a storage policy may specify that certain data should be stored in one or more target cloud storage sites **115A-N**, as described herein.

As another example, a storage policy may specify that a first type of files should be retained for one year in a first

10

target cloud storage site **115A**, that a second type of files should be retained for seven years in a second cloud storage site **115B**, and that a third type of files should be retained indefinitely in a third cloud storage site **115N**. As yet another example, a storage policy may specify that a first type of files (e.g., secondary disk copies needed for rapid disaster recovery) be stored only in storage sites **115**, including cloud storage sites **115A-N**, that can provide sufficient bandwidth, network capacity or other performance to ensure that the time needed to recover a file from the storage device **115** (e.g., cloud storage site **115A-N**) is less a specified recovery time objective.

As another example, a storage policy relating to cloud storage sites **115A-N** may specify that a cloud storage site should be chosen, at least in part, based on the geographical (or network) proximity between a data source (e.g., client **130** and/or secondary storage computing device **165**) and the cloud storage site in order to improve data transfers.

As another example, a storage policy relating to cloud storage sites **115A-N** may specify that a first type of files be stored only on cloud storage sites that have a sufficient level of fault tolerance. For example, a storage policy may specify that a first type of files be stored only on cloud storage sites **115A-N** that replicate copies of their data across two or more geographically separate regions or across two or more separate power grids. As yet another example, a storage policy may specify that a first type of files be stored only on cloud storage sites **115A-N** that satisfy other consumer criteria. For example, a storage policy may specify that a first type of files be stored only on cloud storage sites **115A-N** that are certified as being "environmentally green," that align with particular political or social agendas, that do or do not have operations in certain countries (e.g., sites that do have operations in developing nations and/or do not have operations in embargoed countries), or that satisfy some other consumer criteria.

A storage policy might define different classes of storage that should be utilized for different types of data. For example, a storage policy may define "first-class storage" as rapid access media, such as storage devices having magnetic disk (or faster access) storage media, a high bandwidth network connection to the cloud storage site, and a cloud storage site that satisfies certain performance criteria (e.g., has high bandwidth for faster uploads and/or downloads and/or utilizes RAID or similar methods that improve the fault-tolerance of the site). "Second-class storage" may be defined under a storage policy as a second cloud storage site having magnetic tape (or slower access) data storage, lower bandwidth connections and/or less fault tolerance. As another example, a storage policy may define storage classes based on the actual performance achieved by cloud storage sites or other storage devices **115**. For example, a storage policy may define first-class storage as cloud storage sites that actually achieve a threshold average throughput, data recovery rate, and/or specified error rate.

To facilitate the selection of cloud storage sites on the basis of actual performance, a storage manager **105**, secondary storage computing devices **165** and/or other system components may track, log and/or analyze the performance achieved by cloud storage sites. Thus, a client computer or organization may contract with a cloud storage provider for a defined level of service, where the level of service relates to a storage policy as defined herein (e.g. aggregated data storage volumes, fault tolerance, data recovery rates, threshold latency and/or bandwidth, etc., defined under a service level agreement (SLA).) The client computer may then periodically perform tests or monitor performance of the

US 10,248,657 B2

11

cloud storage provider as compared to the defined level of service to ensure the appropriate level of service.

In some implementations, a storage policy may comprise an audit policy. An audit policy is a set of preferences, rules and/or criteria that protect sensitive data in the storage operation cell 150. For example, an audit policy may define “sensitive objects” as files or objects that contain particular keywords (e.g., “confidential,” or “privileged”) and/or are associated with particular keywords (e.g., in metadata) or particular flags (e.g., in metadata identifying a document or email as personal, confidential, etc.). An audit policy may further specify rules for handling sensitive objects. As an example, an audit policy may require that a reviewer approve the transfer of any sensitive objects to a cloud storage site 115A-N, and that if approval is denied for a particular sensitive object, the sensitive object should be transferred to a local storage device 115 instead. To facilitate this approval, the audit policy may further specify how a secondary storage computing device 165 or other system component should notify a reviewer that a sensitive object is slated for transfer.

In some implementations, a storage policy may comprise a provisioning policy. A provisioning policy is a set of preferences, priorities, rules and/or criteria that specify how various clients 130 (or groups of clients 130, e.g., a group of clients 130 associated with a department) may utilize various system resources, including resources such as available storage on cloud storage sites 115A-N and/or the network bandwidth between the storage operation cell 150 and cloud storage sites 115A-N. A provisioning policy may specify, for example, data quotas for particular clients 130 (e.g., a gigabyte amount of data that can be stored monthly, quarterly or annually). Components of the storage operation cell 150, such as the secondary storage computing devices 165, may enforce the provisioning policy (including quotas) during the transfer of data to secondary storage (e.g., during the process 300, shown in FIG. 3B). If a client (typically associated with a department within an organization) exceeds the policy, then a budget for that client/department may be charged for excess storage or resource allocation.

In some implementations, a storage policy may comprise a cost policy. A cost policy is a set of preferences, priorities, rules and/or criteria that specify how to identify suitable storage locations, including suitable cloud storage locations. For example, a cost policy may describe the method of evaluating a cost function, as described in greater detail herein with respect to FIG. 27. Here again, if a client exceeds the policy, then a budget for that client/department may be charged for excess storage or resource allocation.

A storage policy may be stored in a database of the storage manager 105, such as management index 211, or in other locations or components of the system. As will be described in detail herein, the system may utilize a storage policy when identifying suitable storage locations for various data objects subject to the storage policy.

Additionally or alternatively, a “schedule policy” may specify when and how often to perform storage operations and may also specify performing certain storage operations on sub-clients of data and how to treat those sub-clients. A “sub-client” is a portion of one or more clients 130 and can contain either all of the client’s 130 data or a designated subset thereof. For example, an administrator may find it preferable to separate email data from financial data using two different sub-clients having different storage preferences, retention criteria, etc. A schedule policy may be stored in the management index 211 of the storage manager 105 and/or in other locations within the system.

12

Storage operation cells may contain not only physical devices, but also may represent logical concepts, organizations, and hierarchies. For example, a first storage operation cell 150 may be configured to perform a first type of storage operation such as an HSM operation, which may include backup or other types of data migration, and may include a variety of physical components including a storage manager 105 (or management agent 233), a secondary storage computing device 165, a client 130, and other components as described herein. A second storage operation cell 150 may contain the same or similar physical components; however, it may be configured to perform a second type of storage operation, such as a storage resource management (“SRM”) operation, and may include monitoring a primary data copy or performing other known SRM operations.

Thus, as can be seen from the above, although the first and second storage operation cells 150 are logically distinct entities configured to perform different management functions (e.g., HSM and SRM, respectively), each storage operation cell 150 may contain the same or similar physical devices. Alternatively, different storage operation cells 150 may contain some of the same physical devices and not others. For example, a storage operation cell 150 configured to perform SRM tasks may contain a secondary storage computing device 165, client 130, or other network device connected to a primary storage volume, while a storage operation cell 150 configured to perform HSM tasks may instead include a secondary storage computing device 165, client 130, or other network device connected to a secondary storage volume and may not contain the elements or components associated with and including the primary storage volume. (The term “connected” as used herein does not necessarily require a physical connection; rather, it could refer to two devices that are operably coupled to each other, communicably coupled to each other, in communication with each other, or more generally, refer to the capability of two devices to communicate with each other, often with intervening components in between.) These two storage operation cells 150, however, may each include a different storage manager 105 that coordinates storage operations via the same secondary storage computing devices 165 and storage devices 115. This “overlapping” configuration allows storage resources to be accessed by more than one storage manager 105, such that multiple paths exist to each storage device 115 facilitating failover, load balancing, and promoting robust data access via alternative routes.

Alternatively or additionally, the same storage manager 105 may control two or more storage operation cells 150 (whether or not each storage operation cell 150 has its own dedicated storage manager 105). Moreover, in certain embodiments, the extent or type of overlap may be user-defined (through a control console) or may be automatically configured to optimize data storage and/or retrieval.

The clients 130, as part of their function, may utilize data, which includes files, directories, metadata, and other data objects. The data on the clients 130 is typically a primary copy (e.g., a production copy). During a copy, backup, archive or other storage operation, the clients 130 may send a copy of some data objects to a secondary storage computing device 165 by utilizing one or more data agents 195.

The data agent 195 may be a software module or part of a software module that is generally responsible for storage operations, such as copying, archiving, migrating, and recovering data from client 130 stored in data store 260 or other memory location. Each client 130 may have at least one data agent 195, and the system can support multiple clients 130. Data agent 195 may be distributed between

US 10,248,657 B2

13

client 130 and storage manager 105 (and any other intermediate components), or it may be deployed from a remote location or its functions approximated by a remote process that performs some or all of the functions of data agent 195.

The overall system may employ multiple data agents 195, each of which may back up, migrate, archive, and recover data associated with a different application. For example, different individual data agents 195 may be designed to handle Microsoft Exchange data, Lotus Notes data, Microsoft Windows 2000 file system data, Microsoft Active Directory Objects data, and other types of data known in the art. Other embodiments may employ one or more generic data agents 195 that can handle and process multiple data types rather than using the specialized data agents described above.

If a client 130 has two or more types of data, one data agent 195 may be required for each data type to copy, archive, migrate, and restore the data of the client 130. Alternatively, the overall system may use one or more generic data agents 195, each of which may be capable of handling two or more data types. For example, one generic data agent 195 may be used to back up, migrate, and restore Microsoft Exchange 2000 Mailbox data and Microsoft Exchange 2000 Database data while another generic data agent 195 may handle Microsoft Exchange 2000 Public Folder data and Microsoft Windows 2000 File System data, etc.

The data agents 195 may be responsible for arranging or packing data to be copied, transferred, or migrated into a certain format such as an archive file format. Nonetheless, it will be understood that this represents only one example, and any suitable packing or containerization technique or transfer methodology may be used if desired. Such an archive file may include a metadata list of files or data objects copied in metadata, the file, and data objects themselves. Moreover, any data moved by the data agents may be tracked within the system by updating indexes associated with appropriate storage managers 105 or secondary storage computing devices 165. As used herein, a file or a data object refers to any collection or grouping of bytes of data that can be viewed as one or more logical units.

The network client agent 255 may be a software module, part of a software module, and/or may comprise hardware that generally provides the client 130 with the ability to communicate with other components within the system, such as storage manager 105, other clients 130, and secondary storage computing devices 165. Network client agent 255 may permit communication via one or more proprietary and/or non-proprietary network protocols, notably to cloud-based storage, as described herein.

Generally speaking, the storage manager 105 may be a software module or other application that coordinates and controls storage operations performed by storage operation cell 150. Storage manager 105 may communicate with some or all elements of storage operation cell 150 including clients 130, data agents 195, secondary storage computing devices 165, and storage devices 115 to initiate and manage system backups, migrations, data recovery, and other storage operations.

Storage manager 105 may include a jobs agent 220 that monitors the status of some or all storage operations previously performed, currently being performed, or scheduled to be performed by storage operation cell 150, including storage jobs sent to cloud-based storage. Jobs agent 220 may be communicatively coupled to interface agent 225 (e.g., a software module or application). Interface agent 225 may include information processing and display software, such

14

as a graphical user interface ("GUI"), an application programming interface ("API"), or other interactive interface through which users and system processes can retrieve information about the status of storage operations. Through interface agent 225, users may optionally issue instructions to various storage operation cells 150 regarding the performance of the storage operations as described and contemplated herein. For example, a user may modify a schedule concerning the number of pending snapshot copies or other types of copies scheduled as needed to suit particular requirements. As another example, a user may employ the GUI to view the status of pending storage operations in some or all of the storage operation cells 150 in a given network or to monitor the status of certain components in a particular storage operation cell 150 (e.g., the amount of storage capacity left in a particular storage device 115). In some embodiments, users or other system processes may retrieve information or issue commands by employing API commands sent to the interface agent via the network agent 235.

The storage manager 105 may also include a management agent 233 that is typically implemented as a software module or application program. In general, management agent 233 provides an interface that allows various management agents 233 in other storage operation cells 150 to communicate with one another. For example, assume a certain network configuration includes multiple storage operation cells 150 adjacent to one another or otherwise logically related in a WAN or LAN configuration. In this arrangement, each storage operation cell 150 may be connected to the other through a respective interface agent 225. This allows each storage operation cell 150 to send and receive certain pertinent information from other storage operation cells 150, including status information, routing information, information regarding capacity and utilization, etc. These communications paths may also be used to convey information and instructions regarding storage operations. The storage operation cells 150 can be organized hierarchically such that hierarchically superior cells control or pass information to hierarchically subordinate cells or vice versa.

Storage manager 105 may also maintain a management index 211, database, or other data structure. The data stored in management index 211 may be used to indicate logical associations between components of the system, user preferences, management tasks, media containerization and data storage information or other useful data. For example, the storage manager 105 may use data from management index 211 to track the logical associations between secondary storage computing device 165 and storage devices 115 (or the movement of data as containerized from primary to secondary storage). In the case of cloud-based storage, the management index may indicate which cloud-based storage site(s) stores which data set.

Storage manager 105 may also include a network agent 235 that is typically implemented as a software module or part of a software module. In general, network agent 235 provides the storage manager 105 with the ability to communicate with other components within the system, such as clients 130, data agents 195, and secondary storage computing devices 165. As with the network client agents 255, the network agents 235 may permit communication via one or more proprietary and/or non-proprietary network protocols. Network agent 235 may be communicatively coupled to management light index 245, management index 211, jobs agent 220, management agent 233, and interface agent 225.

Generally speaking, the secondary storage computing device 165, which may include or be a media agent, may be

US 10,248,657 B2

15

implemented as a software module that conveys data, as directed by storage manager **105**, between a client **130** and one or more physical storage devices **115**, such as a tape library, a magnetic media storage device, an optical media storage device, a cloud storage site, or any other suitable storage device. In one embodiment, secondary storage computing device **165** may be communicatively coupled to and control a storage device **115**. A secondary storage computing device **165** may be considered to be associated with a particular storage device **115** if that secondary storage computing device **165** is capable of routing and storing data to that particular storage device **115**.

In operation, a secondary storage computing device **165** associated with a particular storage device **115** may instruct the storage device **115** to use a robotic arm or other retrieval means to load or eject a certain storage media. Secondary storage computing device **165** may also instruct the storage device **115** to archive, migrate, restore, or copy data to or from the storage device **115** or its associated storage media. Secondary storage computing device **165** may also instruct the storage device **115** to delete, sparsify, destroy, sanitize, or otherwise remove data from the storage device **115** or its associated storage media. Secondary storage computing device **165** may communicate with a storage device **115** via any suitable communications path, including SCSI, a Fibre Channel communications link, or a wired, wireless, or partially wired/wireless computer network, including the Internet. In some embodiments, the storage device **115** may be communicatively coupled to the storage manager **105** via a storage area network (SAN).

A secondary storage computing device **165** may also include at least one media file system agent **240**. Each media file system agent **240** may be a software module or part of a software module that is generally responsible for archiving, migrating, restoring, accessing, reading, writing, moving, deleting, sanitizing, or otherwise performing file system and data storage operations on various storage devices **115** of disparate types. For example, media file system agent **240** may be configured to permit secondary storage computing device **165** to open, read, write, close, and delete data on cloud storage sites or storage devices **115** having optical, magnetic, or tape media.

A secondary storage computing device **165** may also include a network agent **235** similar or identical to that described previously. Generally, network agent **235** provides the secondary storage computing device **165** with the ability to communicate with other components within the system, such as other secondary storage computing devices **165**, storage manager **105**, clients **130**, data agents **195**, and storage devices **115**. Network agent **235** generally provides communication via one or more proprietary and/or non-proprietary network protocols.

A secondary storage computing device **165** may also include a content indexing component **205** to perform content indexing of data in conjunction with the archival, restoration, migration, or copying of data, or at some other time. Content indexing of data is described in greater detail herein. Each secondary storage computing device **165** may maintain an index, a database, or other data structure (referred to herein as “secondary storage index” or “SS index” **261**) that may store index data generated during backup, migration, restoration, and other storage operations for secondary storage (“SS”) as described herein, including creating a metadata (MB). For example, performing storage operations on Microsoft Exchange data may generate index data. Such index data provides a secondary storage computing device **165** or other external device with an efficient

16

mechanism for locating data stored or backed up. Thus, an SS index **261** and/or a management index **211** of a storage manager **105** may store data associating a client **130** with a particular secondary storage computing device **165** or storage device **115**, for example, as specified in a storage policy, while an SS index **261**, metadata, database, or other data structure in secondary storage computing device **165** may indicate where specifically the data of the client **130** is stored in storage device **115**, what specific files were stored, and other information associated with storage of the data of the client **130**. In some embodiments, such index data may be stored along with the data backed up in a storage device **115**, with an additional copy of the index data written to index cache in a secondary storage device **165**. Thus the data is readily available for use in storage operations and other activities without having to be first retrieved from the storage device **115**.

Generally speaking, information stored in cache is typically information that reflects certain particulars about operations that have recently occurred. After a certain period of time, this information is sent to secondary storage and tracked. This information may need to be retrieved and uploaded back into a cache or other memory in a secondary computing device before data can be retrieved from storage device **115**. In some embodiments, the cached information may include information regarding the format or containerization of archives or other files stored on storage device **115**.

A secondary storage computing device **165** may also include a deduplication database **297** to perform deduplication of data in conjunction with the archival, restoration, migration, or copying of data, or at some other time. The secondary storage computing devices **165** may also maintain one or more deduplication databases **297**. Single instancing is one form of deduplication and generally refers to storing in secondary storage only a single instance of each data object (or each data sub-object or each data block) in a set of data (e.g., primary data). More details as to single instancing may be found in one or more of the following commonly assigned U.S. patent applications: 1) U.S. Pat. Pub. No. 2006-0224846 (entitled SYSTEM AND METHOD TO SUPPORT SINGLE INSTANCE STORAGE OPERATIONS, U.S. patent application Ser. No. 11/269,512); 2) U.S. Pat. Pub. No. 2009-0319585 (entitled APPLICATION-AWARE AND REMOTE SINGLE INSTANCE DATA MANAGEMENT, U.S. patent application Ser. No. 12/145,342); 3) U.S. Pat. Pub. No. 2009-0319534 (entitled APPLICATION-AWARE AND REMOTE SINGLE INSTANCE DATA MANAGEMENT, U.S. patent application Ser. No. 12/145,347); 4) U.S. Pat. Pub. No. 2008-0243879 (entitled SYSTEM AND METHOD FOR STORING REDUNDANT INFORMATION, U.S. patent application Ser. No. 11/963,623); and 5) U.S. Pat. App. No. 2008-0229037 (entitled SYSTEMS AND METHODS FOR CREATING COPIES OF DATA, SUCH AS ARCHIVE COPIES, U.S. patent application Ser. No. 11/950,376).

Another form of deduplication is variable instancing, which generally refers to storing in secondary storage one or more instances, but fewer than the total number of instances, of each data block (or data object or data sub-object) in a set of data (e.g., primary data). More details as to variable instancing may be found in the commonly assigned U.S. Pat. App. No. 61/164,803 (entitled STORING A VARIABLE NUMBER OF INSTANCES OF DATA OBJECTS). The deduplication module **299** and deduplication database **297** are described in greater detail herein.

US 10,248,657 B2

17

As shown in FIG. 2, clients **130** and secondary storage computing devices **165** may each have associated metabascs or indices (**270** and **261**, respectively). However, in some embodiments, each “tier” of storage, such as primary storage, secondary storage, tertiary storage, etc., may have multiple metabascs/indices or a centralized metabasc/index, as described herein. For example, rather than a separate metabasc or index associated with each client in FIG. 2, the metabascs/indices on this storage tier may be centralized. Similarly, second and other tiers of storage may have either centralized or distributed metabascs/indices. Moreover, mixed architecture systems may be used if desired, that may include a first tier centralized metabasc/index system coupled to a second tier storage system having distributed metabascs/indices and vice versa, etc.

Moreover, in operation, a storage manager **105** or other management module may keep track of certain information that allows the storage manager to select, designate, or otherwise identify metabascs/indices to be searched in response to certain queries as further described herein. Movement of data between primary and secondary storage may also involve movement of associated metadata and index data and other tracking information as further described herein.

In some embodiments, management index **211** and/or SS index **261** may provide content indexing of data generated during backup, migration, restoration, and other storage operations. In this way, management index **211** and/or SS index **261** may associate secondary storage files with various attributes, characteristics, identifiers, or other tags or data classifications associated with the file content. In such embodiments, a user of storage operation cell **150** may search for content within the storage operation cell via the interface agent **225**. Methods of performing content indexing and searching, including collaborative searching, within a storage operation cell **150** are described in the commonly assigned U.S. Patent Publication Nos. 2008-0091655 (entitled METHOD AND SYSTEM FOR OFFLINE INDEXING OF CONTENT AND CLASSIFYING STORED DATA, U.S. patent application Ser. No. 11/694,869) and 2008-0222108 (entitled METHOD AND SYSTEM FOR COLLABORATIVE SEARCHING, U.S. patent application Ser. No. 11/874,122).

In some embodiments, storage manager **105** may also include or be operably coupled to a management light index **245** that may store index data, metadata, or other information generated during backup, migration, restoration, or other storage operations. The management light index **245** provides storage manager **105** and other components with an alternate mechanism for locating data stored or backed up, so that they may more rapidly respond to client **130** or other requests received via HTTP or similar protocols that are susceptible to time-outs.

Management light index **245** may store some subset of the information contained in management index **211**, SS index **261**, client metabasc **270** and/or other information. For example, the management light index **245** comprises the following information about each data file in the storage operation cell **150**: a file name or other descriptor, a descriptor for the client **130** or sub-client associated with the file (typically the client **130** that created the file), the size of the file, the storage location of the file (including the storage device, associated secondary storage computing devices **165** and/or other index data), file type (e.g., file extension or descriptor to associate an application with the file), etc. In some embodiments, the management light index **245** may comprise additional information, such as limited content

18

information. Within the management light index **245**, each data file may also be associated with a token that uniquely identifies the data file. In some embodiments, however, the token may not be unique for all data files in the management light index **245**; instead, the combination of the token with another data field (e.g., the associated client **130**) may be unique.

During the operation of the storage operation cell **150**, management light index **245** may be populated or changed. For example, whenever a secondary storage operation is performed (due to a client **130** request, a scheduled job, the application of a storage policy, or otherwise), the management light index **245** may be updated by the storage manager **105**, secondary storage computing device **165**, or other system component responsible for performing some or all of the storage operation. For example, if a client **130** (or its data agent **195**) requests the creation of a backup, archival, or other secondary copy, the secondary storage computing device **165** (e.g. cloud-based storage site) creating that secondary copy may create one or more new entries in the management light index **245** reflecting the name, location, size, and client **130** associated with the newly created secondary copy. As another example, if due to an ILM storage policy, a file is migrated from a first storage device **115** to a second storage device **115**, a secondary storage computing device **165** may update the management light index **245** to reflect the new location of the file.

In one example, the management light index **245** may only be populated with information regarding data files that originated from clients **130** that connect to the storage operation cell **150** via certain network protocols. For example, the management light index **245** may only be populated with information regarding data files that originated from clients **130** that connect to the storage operation cell **150** via the HTTP protocol.

The secondary storage computing device **165** may include or be operably coupled to a secondary storage light index **247** (“SS light index”). Typically SS light index **247** comprises a subset of the information included in management light index **245**. For example, SS light index **247** includes a subset of information pertaining to secondary storage data files stored in storage devices **115** associated with the secondary storage computing device **165**. During the operation of the storage operation cell **150**, SS light index **247** may be populated or changed in the same or similar manner as management light index **245**.

The management light index **245** and SS light index **247** may be implemented in a non-relational database format, such as C-Tree from Faircom, Inc., SimpleDB from Amazon, Inc., or CouchDB from the Apache Software Foundation. In this way, the storage manager **105** may provide a faster response to client **130** or other requests than if it were to query management index **211**, metabasc **270** and/or SS index **261**, and thus prevent time-outs when communicating via certain network protocols such as HTTP. Components of the storage operation cell **150** system, such as storage manager **105**, may be configured to facilitate data storage provisioning and/or cost charge backs. In some implementations, the system may evaluate the state of stored data relative to enterprise needs by using weighted parameters that may be user defined, e.g., in order to facilitate the generation of or enforcement of a provisioning policy. In some implementations, the system may calculate data costing information and other information including information associated with the cost of storing data and data availability associated with storage operation cells, e.g., in order to facilitate charge backs. The system may identify network

US 10,248,657 B2

19

elements, associated characteristics or metrics with the network elements, receive additional data, such as SRM or HSM data, from storage operation cells, and correlate the additional data with the network elements to calculate a cost of data storage or an availability of data. In some implementations, data may be identified according to user, department, project, or other identifier. In other implementations, data availability or data cost is compared to a service level agreement (SLA). In some implementations, a prediction of media usage is generated according to data use, availability, or cost. Further details regarding provisioning and charge backs may be found in the commonly assigned U.S. application Ser. No. 12/015,470, filed Jan. 16, 2008, entitled “SYSTEMS AND METHODS FOR STORAGE MODELING & COSTING,” which is hereby incorporated herein in its entirety.

In some implementations, storage manager 150 may comprise a management module configured to predict and plan future storage needs. The management module may receive information related to storage activities associated with one or more storage operation components within the storage operation cell under the direction of the storage manager component. The management module is adapted to predict storage operation resource allocations based on the received information related to the storage activities. Further details relating to the prediction of storage operation resource allocations may be found in the commonly assigned U.S. application Ser. No. 11/639,830, filed Dec. 15, 2006, entitled “System and Method for Allocation of Organizational Resources”, and U.S. application Ser. No. 11/825,283, filed Jul. 5, 2007, entitled “System and Method for Allocation of Organizational Resources”, which are hereby incorporated herein in their entirety.

In some implementations, components of the storage operation cell 150, may be configured to copy data of one or more virtual machines being hosted by one or more non-virtual machines (e.g., hosted by a cloud storage site 115A-N). Further details relating to copying data of virtual machines may be found in the commonly assigned U.S. application Ser. No. 12/553,294, filed Sep. 3, 2009, entitled “SYSTEMS AND METHODS FOR MANAGEMENT OF VIRTUALIZATION DATA,” which is hereby incorporated herein in its entirety.

4. Network Agents

Network agent 235 may comprise one or more sub-processes or network subagents, which are typically implemented as a software module or part of a software module. Each network subagent may be responsible for managing communications between the network agent 235 and a remote device conducted via a particular network protocol, such as HTTP. Remote devices might include any component of the storage operation cell 150, such as clients 130, secondary storage computing devices 165, storage devices 115, storage managers 105 or other networked devices. Each network subagent may do some or all of the following: accept or initiate connections to remote devices; authenticate remote devices and/or specific users on remote devices; receive requests from remote devices; provide responses to remote devices; log requests and responses; detect or respond to network time-outs; compress or encrypt data; serve data or content to remote devices; redirect remote devices to other system components; call other applications, scripts, or system resources; and implement bandwidth throttling. Each network subagent may include instructions for interpreting routines, data structures, object classes, and/or protocols defined in a particular API or similar interface.

20

Typically, each subagent manages communications made via a particular network protocol. For example, each subagent manages communications utilizing a particular layer protocol, such as a transport layer protocol like Transport Control Protocol (“TCP”) from the TCP/IP (Internet Protocol). However, a subagent may additionally or alternatively manage one or more protocols from a layer other than the transport layer (e.g., application layer), more than one transfer layer protocol.

Typical network subagents, include an HTTP subagent, an FTP subagent, and a proprietary protocol subagent. An HTTP subagent may manage connections that utilize HTTP and/or HTTP over TLS/SSL (“HTTPS”). An FTP subagent may manage connections to the network agent 235 that utilize the FTP and/or secure FTP. A proprietary protocol subagent may manage connections that utilize a particular proprietary application-layer protocol. In some embodiments, the proprietary protocol subagent may be configured to facilitate a virtual private network connection running over an HTTPS protocol, or another type of open/secure pipe wrapped in an HTTPS protocol. Non-exclusive examples of other possible network subagents (not shown) include network subagents to implement the common internet file system (CIFS) protocol and the network file system (NFS) protocol.

5. Network Client Agents

Network client agents 255 are similar to the network agents 235. Typically, each network client subagent manages communications utilizing a network protocol, and is substantially similar to the network subagents described above. Thus, typical network client subagents include an HTTP client subagent, an FTP client subagent, a proprietary protocol client subagent, and a telecommunications protocol client subagent. An HTTP client subagent may be a web browser application configured to connect both to network client agents 255 as well as other resources such as general Internet or web servers. A telecommunications protocol client subagent may manage remote connections that utilize data transfer protocols supported by certain types of telecommunications networks, e.g., Global System for Mobile (GSM), code/time division multiple access (CDMA/TDMA), and/or 3rd Generation (3G) telecommunications networks. For example, telecommunications protocol client subagent may permit a user to initiate an HTTP connection by using an API associated with a mobile operating system such as Windows Mobile, BlackBerry OS, iPhone OS, Palm OS, Symbian, and Android.

6. Media File System Agent

Media file system agent 240 may comprise one or more media submodules. Each media submodule may permit the media file system agent 240 to perform basic file system commands (e.g., open, read, write, close, and delete) on a certain type of storage device 115, and/or to otherwise direct a certain type of storage device 115 to perform file system or storage operations. For example, the media file system agent 240 may comprise tape, optical and/or magnetic submodules to open, read, write, close, and delete data files on storage devices utilizing tape, optical and magnetic media, respectively. Media file system agent 240 may also comprise one or more cloud storage submodules 236 that permit the media file system agent 240 to open, read, write, close, and delete data files stored on cloud storage sites and/or otherwise direct cloud storage sites to perform data storage operations.

US 10,248,657 B2

21

7. Cloud Storage Submodules: Vendor-Agnostic File System Calls, Buffering of Storage Requests, and Logging Cloud Storage Performance

Each cloud storage vendor associated with a particular cloud storage site **115A-N** utilized by the system may provide an API that has vendor-specific implementation of basic file system calls. For example, each vendor API may prescribe a different functional call for opening/creating a new data file on the vendor's cloud storage site. Typically a cloud storage vendor API will utilize REST-based protocols. The system described herein may use a cloud storage submodule to map each generic file system command (e.g., an open command) to the various implementations of the command as defined in each of the APIs provided by the various cloud storage vendors. Using the mapping, a cloud storage submodule may convert a generic file system command received by the media file system agent **240** into the appropriate vendor-specific call for a target cloud storage site **115A-N**. In this way, the cloud storage submodule permits the system to ignore implementation details of the various cloud storage sites **115A-N** used by the system and simply treat each site in a manner analogous to local data storage media, such as local optical or tape media. In this manner, a cloud storage submodule may obviate the need for complex scripting or the addition of disparate cloud gateway appliances to write data to multiple cloud storage site targets. In this way, a cloud storage submodule **236** also presents clients **130** and other system components with a unified name space, even if the system is storing data on multiple cloud storage sites **115**.

For example, the cloud storage submodule **236** includes an interface to translate the REST-based protocols of the Amazon S3 APIs, the Windows Azure APIs and the Rack-space APIs into generic commands for use with a file system such as Windows, Solaris, Unix or Linux. Thus, the cloud storage submodule converts the format and parameters of relevant storage vendor APIs, such as "open file" and "write file", into a normalized or generic format for use with file systems. (The cloud storage submodule may likewise convert, if needed, the generic format into a format for specific file systems such as Windows, Linux, etc.) As shown in FIG. 2, the cloud storage submodule **236** may reside on media file system agent **140** located on the secondary storage computing device **165** to initiate file system and storage operations on cloud storage sites (including data transfers to and from a site). To initiate file system and storage operations, the cloud storage submodule **236** may invoke the network agent **235**, via an HTTP subagent, an FTP subagent, or another type of network subagent, to open a suitable network connection to a target cloud storage site so that the cloud storage submodule may make various file system requests upon the target cloud storage site for storage operations via this network connection.

Some cloud storage site APIs may provide advanced functionality to manipulate files stored on a cloud storage site that extend beyond basic file system calls such as open, read, write. For example, cloud storage site APIs may provide commands for the encryption, compression and/or other advanced file operations. Cloud storage submodules may map generic advanced file operations (e.g., a generic encryption command) to the various implementations of the command as defined in each of the APIs provided by the various cloud storage vendors. As one example, a cloud storage site API may provide a command to encrypt a file located on the cloud storage site using an encryption method that does not result in the cloud storage site receiving a key (or does not result in the cloud storage site receiving or

22

retaining other information sufficient to decrypt an encrypted file). For example, a cloud storage site API may permit storing encrypted data belonging to a client on a cloud storage site, together with an encrypted version of the encryption key that was used to encrypt the encrypted data. A password would be required from the client in order to decrypt the encrypted version of the encryption key stored on the storage system belonging to the application service provider. This is advantageous for the client, because it would prevent the application service provider from decrypting the data belonging to the customer, without the customer's permission.

Additionally, using the mapping, a cloud storage submodule **236** may permit other system components to direct one cloud storage site **115** to transfer some or all files to another cloud storage site **115**, without first transferring the files back to the storage cell **150**. In this way, the system may efficiently and effectively "fire" underperforming or expensive cloud storage sites **115** or otherwise adjust how it uses multiple cloud storage sites **115A-N**. For example, if the system determines that a cloud storage site is underperforming, it may transfer files from the underperforming site to a different site that is meeting performance metrics specified in a storage policy.

When a cloud storage submodule **236** initiates file system and storage operations on a cloud storage site, it may determine or test and record (or report, e.g., to a storage manager **105**) the performance achieved by the cloud storage site, such as the throughput of the site, the number of failures that occurred, the number of timeouts, speed of restores, speed of responses to queries, or other metrics. By determining the actual performance of cloud storage sites **115A-N**, the storage operation cell **150** may adjust its classifications of various cloud storage sites **115** (e.g., as first-class storage, as second-class storage, etc.) dynamically or periodically. Additionally, on a periodic basis, the system may determine which cloud storage sites are underperforming so that it may transfer files from the underperforming site to a different site that is meeting performance metrics specified in a storage policy or take other suitable action (e.g., requesting a reduced storage price).

A cloud storage submodule **236** may also store and/or manage credentials or other authorization and connection information (e.g., site configuration settings, login information, certificates, etc.) that permit the cloud storage submodule to perform storage operations on a cloud storage site **115**. To add a new cloud storage site **115** to the storage operation cell **150**, the system may populate each cloud storage submodule with the appropriate configuration settings or credentials for the new site.

The cloud storage submodule **236**, during a period of its operation, may receive a series of similar requests for the submodule to transfer data to a target cloud storage site (e.g., cloud storage site **115A**); each individual request in the series may only involve a small amount of data (e.g., a few data blocks or a small data object such as an email). For example, since the system may utilize cloud storage submodule to transfer data to cloud storage sites **115A-N** during containerized deduplication, it may receive a series of similar file requests (e.g., to write several small email data objects to the same target container file on the same target cloud storage site). To facilitate more efficient data transmission, which may occur over a lossy and/or latent WAN (such as the Internet), the cloud storage submodule may utilize two or more local buffers (e.g., buffers stored in local memory, such as local RAM) to manage the series of transfer requests. The buffers need not be large, and could be

US 10,248,657 B2

23

set in one embodiment to 128 k each, although larger buffers may of course be used, and the size of the various buffers used by the cloud storage submodule may be configurable by the user.

As an example, the cloud storage submodule **236** may maintain a first buffer that reflects the data transmitted in the last storage request from the cloud storage submodule to the target cloud storage site **115A**. By maintaining the first buffer, the cloud storage submodule can easily and more quickly restart data transmission if the last request fails (e.g., due to packet loss/latency). In this example, the cloud storage submodule may maintain a second buffer that aggregates the data associated with various storage requests received by the cloud storage submodule from other system components (e.g., the deduplication module **299**) since the cloud storage submodule began transmitting the last storage request to the target cloud storage site **115A**. In this example, the contents of the second buffer may be sent as a second request to the cloud storage site **115A** once the cloud storage submodule successfully transmits the last request and/or receives confirmation that the cloud storage site **115A** successfully received the last request.

In this example, the size of the buffers may be adjusted to reflect relative network latency and network bandwidth. For example, a larger buffer size may be chosen if the network latency is high, so that more data may be added to the second buffer while the cloud storage submodule transmits the last request and/or awaits a response from the target cloud storage site **115A** regarding the last storage request. As another example, a smaller buffer size may be chosen if the network bandwidth is low, since the maximum transmission size imposed by TCP/IP protocols may be lower. Buffering a series of requests in this manner may improve transmission efficiency, since it may result in the transmission of less non-data (e.g., less transmission of padding zeros added to the transmission as a result of TCP/IP protocols).

FIG. 3A is a flow diagram illustrating a method **300** for writing data to cloud storage sites. A cloud storage submodule **236** or another system component may perform method **300** to provide other system components with vendor-agnostic file system calls and/or efficient data transmission to cloud storage sites **115A-N**. At step **340**, cloud storage submodule **236** receives a file system request to write data to a target cloud storage site **115A-N**. For example, cloud storage submodule **236** may receive a request to write **N** blocks to a first container file located on a first cloud storage site. At step **350**, cloud storage submodule **236** adds the received data (e.g., **N** blocks of data) to a buffer.

Although not shown, prior to step **350**, cloud storage submodule **236** may first determine if the received request has sufficiently similar characteristics to other prior requests that are reflected in the buffer. For example, cloud storage submodule **236** may determine if the instant file system request has the same target file on the same target cloud storage site **115A-115N** as other file system requests whose data is already stored in the buffer. If the request is not sufficiently similar, cloud storage submodule **236** may proceed to step **370** instead. Cloud storage submodule **236** may also allocate a new buffer and initiate a new parallel process **300** to handle the latest request using the new buffer. Additionally, although not shown, prior to step **350**, cloud storage submodule **236** may determine if the file system request relates to a set of data exceeding the buffer size (or another threshold size). If the related set of data is larger than the threshold size, the cloud storage submodule **236** may simply convert the received file system request to one or more vendor-specific API calls and transmit the set of data

24

separately from the other buffered requests before proceeding to step **340**. For example, a received 2 MB file may bypass the buffering and simply proceed on in the process.

At decision step **360**, cloud storage submodule **236** determines if the buffer is full. If it is not full, steps **340-360** are repeated. For example, cloud storage submodule **236** may receive a request to store **M** additional blocks to the same file and add these **M** blocks of data to the buffer. If the buffer is full at decision step **360**, cloud storage submodule **236** proceeds to step **370**. At step **370**, cloud storage submodule converts the received file system requests to one or more vendor-specific API calls. For example, using the mapping described herein, cloud storage submodule may identify the calls from the target cloud storage site API that cause the target cloud storage site to (1) open a target file on the target cloud storage site for writing, and (2) write the received and buffered data to the target file. At step **380**, cloud storage submodule transmits the buffer using the vendor-specific API calls. To transmit the buffer, cloud storage submodule may utilize a network agent **235** to establish an HTTP, HTTPS, and/or other suitable network connection to the target cloud storage site. At step **390**, generally after waiting a sufficient time for a response from the target cloud storage site, cloud storage submodule determines if the transmission was successful. If it was successful, process **300** returns. Otherwise, steps **380** and **390** are repeated and the data is re-transmitted.

Although not shown in FIG. 3A, while cloud storage submodule **236** is performing steps **380-390**, it may also allocate a new buffer to manage new file system requests and may initiate a parallel process **300** to manage these new file system requests using the new buffer.

Cloud storage submodule **236** may be configured to permit a direct interface to cloud storage sites **115A-N** by presenting cloud storage sites to a user or system in the same manner as a local storage volume. For example, a cloud storage submodule **236** operating on a computing device may permit the operating system of that computing device to "mount" a cloud storage site as a storage volume or otherwise provide an interface to have the cloud storage site display to the operating system of the computer as a locally attached drive (similar to network attached storage (NAS)). Cloud storage submodule **236** may further permit the operating system to make various file system requests upon the mounted cloud storage site in a manner analogous to local disk storage. In such implementations, cloud storage submodule **236** may be installed on clients **130** to facilitate easier utilization of remote cloud storage sites.

8. Migrating or Copying Data to Secondary Storage, Including Secondary Cloud Storage

FIG. 3B shows a flow diagram illustrating a suitable routine **300** for migrating or copying data into an archive format in secondary storage, including secondary cloud storage. In step **310**, the system receives a copy of an original data set from a file system. Alternatively, the system may access the copy or otherwise communicate with data storage components in a data storage system to gain access to the data to be copied.

At step **310** (or at any other suitable point in routine **300**), the system may check the original data set against any audit policies applicable to the data set to determine if the data set comprises one or more sensitive objects and whether the migration or copying of sensitive objects to secondary storage requires approval by a reviewer or other action. If approval or other action is required, the system may take appropriate steps in accordance with the applicable audit policy, such as notifying a reviewer of the sensitive object

US 10,248,657 B2

25

and pausing the routine **300** until the system receives an indication that the reviewer approves of the migration/copying. As another example, the system may continue to perform routine **300**, but only for the non-sensitive data objects in the data set. If the system receives an indication that the reviewer does not approve of the migration/copying of a sensitive object, the system may take other steps in accordance with the applicable audit policy. For example, the system may break the set into two or more data subsets (one containing no sensitive objects) and store the data subsets that have sensitive objects in an archive format at a suitable alternative secondary storage location (e.g., a local storage device **115**).

In step **320**, the system may index the data in the copy. For example, the system may index the content of the data as described herein. In step **330**, the system may perform deduplication upon the data, by removing duplicate instances of files, data objects, blocks, sub-objects, and other information, and storing deduplicated data (or “dehydrated data”) in secondary cloud storage, typically in an archive file format. Although not shown explicitly, in some embodiments, the indexing of the data at block **320** may occur after deduplication of the data at block **330**, in order to reduce the volume of data that the system must index. Indexing, deduplication, and storing deduplicated data for cloud storage are described in greater detail herein, beginning with deduplication and followed by indexing.

Although not shown, the system may encrypt the data before or after a secondary copy or archival copy is created. For example, the system may employ many different techniques for encrypting the archive copy, including encryption techniques that satisfy Federal Information Processing Standards (FIPS). Further details about encryption and encrypting archive copies of data may be found in commonly assigned U.S. Patent Publication No. US2008-0320319A1, filed on Mar. 31, 2008, entitled SYSTEM AND METHOD FOR ENCRYPTING SECONDARY COPIES OF DATA (U.S. patent application Ser. No. 12/060,026). Additionally, although not shown, the system may compress the data before or after a secondary copy or archival copy is shown. For example, the system may employ many different well-known techniques or applications for compressing data, including Lempel-Ziv (LZ) techniques, DEFLATE techniques, and LZ-Renau (LZR) techniques.

In some implementations, the techniques described herein may be utilized to make secondary disk copies to disaster recovery (DR) locations using auxiliary copy or replication technologies as noted above.

In some examples, the techniques described herein may be used on copies of data created by replication operations such as CDR (Continuous Data Replication) and DDR (Discrete Data Replication). For example, for data protected by a replication operation, multiple Consistent Recovery Points (CRPs) are established, and the replicated data can be analyzed at such CRPs. To create a CRP, the system suspends writes to the data, and makes a copy of the data. The system then transfers that copy to another location, such as to one of the cloud storage sites. Further details on CDR may be found in the assignee’s U.S. Pat. No. 7,651,593, entitled “SYSTEMS AND METHODS FOR PERFORMING DATA REPLICATION”.

9. Deduplication

Referring to FIG. 4, the deduplication module **299** includes various components that perform various functions associated with deduplication, some of which are described below. More details may be found in the assignee’s U.S. Pat. Pub. No. 2008-0243958, entitled SYSTEM AND METHOD

26

FOR STORING REDUNDANT INFORMATION (U.S. patent application Ser. No. 12/058,367), the entirety of which is incorporated by reference herein. These components include a data object identification component **410**, an identifier generation component **420**, an identifier comparison component **425**, and a criteria evaluation component **430**. The data object identification component **410** identifies files, data objects, sub-objects, or blocks, such as in response to a storage operation. The identifier generation component **420** generates an identifier for the file, data object, sub-object, or block (identifiers are discussed in more detail below). The identifier comparison component **425** performs comparisons of identifiers of various files, data objects, sub-objects, or blocks to determine if the files, data objects, sub-objects, or blocks contain similar data (for example, the identifier comparison component **425** can compare identifiers of two or more files, data objects, sub-objects, or blocks to determine if the files or data objects contain the same data, metadata such as access control lists (ACLs), descriptive metadata that describes the files, data objects, sub-objects, or blocks (e.g., file name, file size, file author, etc.) of the two or more files, data objects, sub-objects, or blocks). The criteria evaluation component **430** evaluates aspects of files, data objects, sub-objects, or blocks against a set of criteria. The deduplication module **299** may also contain other components that perform other functions.

Examples of identifiers include a hash value, message digest, checksum, digital fingerprint, digital signature, or other sequence of bytes that substantially uniquely identifies the file or data object in the data storage system. For example, identifiers could be generated using Message Digest Algorithm 5 (MD5) or Secure Hash Algorithm SHA 512. In some instances, the phrase “substantially unique” is used to modify the term “identifier” because algorithms used to produce hash values may result in collisions, where two different data objects, when hashed, result in the same hash value. However, depending upon the algorithm or cryptographic hash function used, collisions should be suitably rare and thus the identifier generated for a file or data object should be unique throughout the system. The term “probabilistically unique identifier” may also be used. In this case, the phrase “probabilistically unique” is used to indicate that collisions should be low-probability occurrences, and, therefore, the identifier should be unique throughout the system. In some examples, data object metadata (e.g., file name, file size) is also used to generate the identifier for the data object.

The hash values may also be used to verify data transferred to a cloud storage site. For example, a file may first be locally hashed at a client to create a first hash value. The file may then be transferred to the cloud storage site. The cloud storage site in turn similarly creates a hash value and sends this second hash value back. The client may then compare the two hash values to verify that the cloud storage site properly received the file for storage. As explained herein, various system components, from the client, to storage cell components, to cloud gateways, to cloud storage sites themselves may perform such hashing and generation of hash values for verification.

10. Object-Level Deduplication

The deduplication module **299** may conduct object-level deduplication as follows before transferring data to cloud storage sites **115**. (Further details may be found in the assignee’s U.S. Pat. Pub. No. 2009-0319585, entitled APPLICATION-AWARE AND REMOTE SINGLE INSTANCE DATA MANAGEMENT (U.S. patent application Ser. No. 12/145,342).) First, the deduplication module **299** generates an identifier for a data object. After generating

US 10,248,657 B2

27

the identifier for a data object, the deduplication module 299 determines whether it should be stored to the cloud storage site 115 as a secondary copy (e.g., a backup copy) of the data of the clients 130. To determine this, the deduplication module 299 accesses the deduplication database 297 to check if a copy or sufficient number of copies or instances of the data object have already been appropriately stored on a cloud storage site 115. The deduplication database 297 utilizes one or more tables or other data structures to store the identifiers of the data objects that have already been stored on a cloud storage site 115. In one implementation, the system may store multiple copies of a data object, but only one copy of the data object with each of multiple, different cloud storage sites, and the data structure described herein facilitates that process.

If an insufficient number of copies or instances of the data object have already been appropriately stored on a cloud storage site 115, the deduplication module 299 sends the data object to one of the cloud storage site 115 for storage and adds its identifier to the deduplication database 297 (or if an instance already existed, the deduplication module 299 may add a reference, e.g., to an index in the deduplication database 297, such as by incrementing a reference count in the index). The deduplication module may also store in the deduplication module 297 a URL, link, path or identifier of the location or identity of the particular cloud storage site if multiple sites are being used.

If a sufficient number of instances have been appropriately stored, the deduplication module 299 can avoid sending another copy to the cloud storage site 115. In this case, the deduplication module 299 may add a reference (e.g., to an index in the deduplication database 297, such as by incrementing a reference count in the index) to the already stored instance of the data object, and may only store a pointer to the data object on the cloud storage site 115. The link or pointer may comprise a URL to a data object or file within a cloud storage site 115A-N. As explained below, adding a reference to the already stored instance of the data object enables the storage of only a single instance of the data object (or fewer instances of the data object) while still keeping track of other instances of the data object that do not need to be stored.

In some examples, instead of the clients 130 sending the data objects to the deduplication module 299 and the deduplication module 299 generating the identifiers, the clients 130 can themselves generate an identifier for each data object and transmit the identifiers to the deduplication module 299 for lookup in the deduplication database 297. This example may be useful if the clients were to send data directly to the cloud storage site 115, and thus deduplicating data before sending it can conserve time and bandwidth, and storage resources at the cloud storage site (which may charge based on amount of data stored.) If the deduplication module 299 determines that a sufficient number of instances of a data object have not already been appropriately stored on a cloud storage site 115, the deduplication module 299 can instruct the client 130 to send it a copy of the data object, which it then stores on the cloud storage site. In this example, the deduplication module may reside on a server to which the client is connected (e.g. over a LAN or secure WAN). Alternatively, the client 130 itself can send the copy of the data object to the cloud storage site 115, in which case the client may have the deduplication module 299 residing on the client. In some examples, the deduplication module 299 generates the identifier on data already stored on the cloud storage site 115 or on other cloud storage sites (e.g., secondarily stored data is deduplicated).

28

The deduplication module 299 can support encrypted data objects. For example, one client 130 could generate an identifier for a data object, and then encrypt it using one encryption algorithm. Another client 130 could generate an identifier for another data object, and then encrypt it using another encryption algorithm. If the two data objects are identical (meaning the two objects have the same data, while their metadata, such as ACLs or descriptors, could be different), they will both have the same identifier. The deduplication module 299 can then store both encrypted instances of the data object or only a single encrypted instance (or a reduced number of encrypted instances). In some examples, the deduplication module 299 stores a key or other mechanism to be used to encrypt and/or decrypt data. The deduplication module 299 can also support compressed data objects. In general, the same compression algorithm may be used to compress data objects. Therefore, the deduplication module 299 can generate an identifier for a data object before or after it has been compressed.

11. Data Structures for Object-Level Deduplication

Some details will now be provided of suitable object, sub-object level and block level deduplication that the system may employ. Further details may be found in the assignee's U.S. patent application Ser. No. 12/565,576, filed Sep. 23, 2009, entitled "Systems and Methods for Managing Single Instancing Data" and the assignee's U.S. patent application Ser. No. 12/553,199, filed Sep. 3, 2009, entitled "TRANSFERRING OR MIGRATING PORTIONS OF DATA OBJECTS, SUCH AS BLOCK-LEVEL DATA MIGRATION OR CHUNK-BASED DATA MIGRATION". FIGS. 5A and 5B are block diagrams illustrating various data structures which aspects of the invention may utilize for deduplicating and storing copies or instances of data objects on the cloud storage site 115. FIG. 5A illustrates a data structure 500 used in a storage operation. For the storage operation, a chunk folder 502 is created on the cloud storage site 115. Contained within the chunk folder are three files: 1) a metadata file 504; 2) an "N" file 506; and 3) a single instance, or "S" file 508. The three files are each logical containers of data. The "S" file stores deduplicated data (e.g., deduplicated files). The "N" file stores data that is not deduplicated (e.g., metadata, such as descriptive metadata associated with deduplicated files). The metadata file stores references to the location(s) of data objects in the "S" file and the "N" file. Note that although three container files are shown (S, N, and index), in some embodiments a chunk folder may comprise more than one "S" file (e.g., S1, S2 . . . Sy, where y is an integer) to store deduplicated data and/or more than one "N" file (e.g., N1, N2 . . . Nz, where z is an integer). While described as being stored on the cloud storage site 115, the "N" and metadata files may alternatively or additionally be stored elsewhere, such as on the secondary storage computer device 165 and/or storage manager 105.

The chunk folder 502 and the files 504-508 may be equivalent to a directory and files (or folder and files) on a file system. For example, the chunk folder 502 may be a directory and the files 504-508 may be files located within the directory. As another example, the chunk folder 502 may be a file and the files 504-508 may be portions of the file. As another example, the files 504-508 may be collections of blocks or bytes grouped together. Those of skill in the art will understand that the chunk folder 502 and the files 504-508 may be comprised in various data structures and are not limited to a directory and files within the directory.

The deduplication module 299 places data objects in the "S" file 508 that meet certain criteria for deduplication.

US 10,248,657 B2

29

These criteria may include the following: 1) that the data object has been determined to be data or of type data (as opposed to metadata or of type metadata); and 2) that the data object is larger than a pre-configured size, such as 64 Kb. Type data is generally the payload portion of a file or data object (e.g., a file's contents) and type metadata is generally the metadata portion of the file or data object (e.g., metadata such as file name, file author, etc.). This pre-configured size may be configurable by an administrator or other user with the appropriate permissions. For example, if the administrator wants all data objects of type data to be deduplicated, the administrator can set the pre-configured size to 0 Kb. As another example, if the administrator wants only data objects of type data greater than 128 Kb to be deduplicated, the administrator can set the pre-configured size to 128 Kb.

The deduplication module 299 determines if a data object meets these criteria by evaluating aspects of the data object (e.g., its type, its size) against the criteria. If so, the deduplication module determines if a sufficient number of instances of the data object have already been appropriately stored on the cloud storage site 115 (or elsewhere), which the deduplication module determines by generating or retrieving an identifier for the data object and looking up the identifier in the deduplication database 297. During this lookup, to determine whether other instances were appropriately stored, the deduplication database 297 may restrict the lookup to only those instances of the object stored on certain cloud storage sites 115 and/or certain classes of cloud storage sites 115. For example, the deduplication database 297 may restrict the lookup to those cloud storage sites 115 that would satisfy applicable storage policy parameters, such as class of storage used for the object. Additionally, during this lookup, the deduplication database 297 may restrict the lookup to only those instances of the object stored within a certain time frame. For example, the deduplication database 297 may restrict lookup only to those instances stored within secondary storage in the last seven years.

If a sufficient number of instances of the data object have already been appropriately stored on a cloud storage site 115, the deduplication module 299 places the data object in the "S" file 508. The deduplication module 299 may also apply other criteria that the data object must meet for deduplication (e.g., criteria based upon characterizing or classifying the data object using techniques such as those described in commonly assigned U.S. Pat. Pub. No. 2007-0185925 (entitled SYSTEMS AND METHODS FOR CLASSIFYING AND TRANSFERRING INFORMATION IN A STORAGE NETWORK, U.S. patent application Ser. No. 11/564,119), the entirety of which is incorporated by reference herein).

For each data object that is placed in the "S" file 508, the deduplication module 299 adds a reference to the data object in the metadata file 504, called an internal reference. For example, the internal reference may be a pointer or link to the location of the data object in the "S" file 508. As further described herein, the deduplication module 299 maintains a primary table that contains all the deduplication records of all data objects for which an identifier was created. The deduplication module 299 may add as the internal reference a record of the already stored instance of the data object from the primary table.

The deduplication module 299 places data objects in the "N" file 506 that do not meet the above criteria for deduplication. For example, a data object may be metadata (e.g., ACLs for a file that is placed in the "S" file, file descriptor information, etc.). In this case, the data object will be placed

30

in the "N" file. As another example, a data object may be smaller than the pre-configured size, e.g., the data object is smaller than 64 Kb. In this case, the deduplication module 299 may incur too much overhead to generate its identifier and perform a lookup of the identifier in the deduplication database 297. Therefore, the data object is placed in the "N" file. As another example, a prior instance of an object may have been stored on tape and reflected in the deduplication database 297, but the storage policy applicable to the current data object requires disk storage. Therefore, the data object is placed in the "N" file 506. For each data object that is placed in the "N" file 506, the deduplication module 299 may also add a reference to the data object in the metadata file 504, called an internal reference. For example, the internal reference may be a pointer or link to the location(s) of the data object in the "N" file. A new "N" file may be created during each storage operation job.

FIG. 5B illustrates a data structure 510 that may be created as a result of one or more storage operations. The data structure 510 is similar to the data structure 500 illustrated in FIG. 5A, but now includes a second chunk folder 502'. For example, the deduplication module 299 may create the second chunk folder 502' as a result of a second storage operation. Consider the situation where a single data object is subjected to two successive storage operations. The first storage operation would result in the creation of the first chunk folder 502 illustrated in FIG. 5A, with the single data object in a first "S" file 508, its metadata (e.g., ACLs) in a first "N" file 506, and any references to the single data object and its metadata in a first metadata file 504.

The second storage operation would result in the creation of the second chunk folder 502' illustrated in FIG. 5B. As illustrated in FIG. 5B, the second chunk folder 502' would have a second "N" file 506 containing the metadata (e.g., the ACLs of the single data object, regardless of whether they have changed) and a second metadata file 504. Instead of having a second "S" file 508, the second metadata file 504 would have a pointer 515 to the single data object contained in the first "S" file 508. Because an instance of the single data object is already contained within the first "S" file 508, there is no need for another instance of it to be contained within the second "S" file 508. However, there is a need to keep a record of the fact that the second storage operation involved an instance of the single data object. This is accomplished by the pointer 515 within the second metadata file 504.

In some cases, instead of always placing in the "N" file 508 data objects that do not meet the above criteria for deduplication, the deduplication module 299 generates an identifier for the data object, looks up the identifier in the deduplication database 297 to see if the data object has already been stored, and if not, places it in the "S" file 508. If the data object has already been stored, the deduplication module would then add a pointer to the location of the instance of the previously stored data object in the metadata file 504. For example, this variation on the process could be used to deduplicate metadata instead of always storing it in the "N" file 506.

FIG. 5C illustrates a data structure 520 for the metadata file 504. The data structure 520 consists of one or more stream headers 522 and stream data 524. The stream header 522 describes a data object contained in an "N" file 506 or an "S" file 508 (e.g., its location, its size, an offset within the file, etc.). The stream data 524 contains the pointer to the data object contained in the "N" file 506 or the "S" file 508. For example, the pointer may give its location within the "N" file 506 or the "S" file 508. The location of the data

US 10,248,657 B2

31

object may be given by offsets within the “N” file 506 or the “S” file 508. For example, its location may be given by a starting offset, and its length or size. As another example, its location may be given by a starting offset and an ending offset. As previously mentioned, the data object may be in an “S” file 508 in another chunk folder, and the stream data 524 would point to this “S” file in the other chunk folder (e.g., give its location in the “S” file in the other chunk folder). Each time the deduplication module 299 places a data object in the “S” file 508, the deduplication module 299 adds a stream header 522 and corresponding stream data 524 to the metadata file 504.

One advantage of the data structures 500, 510, 520 illustrated in FIGS. 5A through 5C and the techniques described herein is that they reduce the number of files stored on the file system of the cloud storage site 115. Thus, there are as little as three files created for each storage operation—the metadata file 504, the “N” file 506, and the “S” file 508. Therefore, a maximum number of files on the file system of the cloud storage site 115 may be as low as the number of storage operations performed by the deduplication module 299 multiplied by three. File systems of certain operating systems may have practical limits to the numbers of files that they can store that are well below their theoretical limits. For example, a file system may not, in practice, be able to store a number of files above a certain threshold without experiencing significant system degradation (which can be defined in numerous ways, such as an increase in seek time of randomly accessed media that is ten percent longer than normal, a delay in reads or writes on randomly accessed media, or in other ways).

By storing multiple data objects in a small number of container files (as few as two), the storing of each data object as a separate file on the file systems of the cloud storage site can be avoided. This reduces the number of files that would be stored on the file systems of the cloud storage site, thereby ensuring that the cloud storage site can adequately store the data of computing devices in the data storage network. Therefore, the file system of the cloud storage site may not necessarily have to contend with storing excessively large numbers of files, such as millions of files or more. Accordingly, these techniques enable very large numbers of data objects to be stored without regard to the limitations of the file system of the cloud storage site.

Further, separate files may be established for separate customers using the cloud storage site. So, the cloud storage site 115A may establish separate folders for each new customer who contracts to store data at the site, and thus that customer’s data is logically segregated from data of other customers.

Even if the deduplication module 299 performs numerous storage operations using these data structures 500, 510, this will result in far fewer files on the cloud storage site 115 than storage operations where each involved data object is stored as a separate file. Another advantage is that the metadata files 504 could be used to replicate the data stored in the deduplication database 297 or reconstruct the deduplication database 297 if its data is ever lost or corrupted. This is because the metadata files 504 may store essentially the same information as what is stored in the deduplication database 297.

However, the storage of data objects in containers such as the “N” file 506 and the “S” file 508 may create additional complexities when it comes time to prune or delete data objects involved in previous storage operations. This is because the data objects are not stored as files on the file system and thus cannot be directly referenced by the file

32

system. For example, consider a first storage operation, involving a first file and a second file, and a second storage operation, involving the first file and a third file, both occurring on the same day. Further consider that the first storage operation’s files are eligible to be pruned after 15 days and the second storage operation’s files are eligible to be pruned after 30 days. Using the techniques described herein, the first storage operation would store the first and second files in an “S” file 508 and the second storage operation would store a pointer to the first file in an “N” file 506 and the third file in another “S” file 508.

After 15 days have elapsed, the first and second files are eligible to be pruned. The first file is referenced by the “N” file 506 of the second storage operation and cannot yet be pruned. However, the second file, because it is not referenced by any “N” files 506 in any other storage operations, can be pruned. Using the metadata file 504 corresponding to the “S” file 508, the deduplication module 299 locates the second file within the “S” file 508. The deduplication module 299 can then instruct the operating system (e.g., a Windows operating system, a Unix operating system, a Linux operating system, etc.) of the cloud storage site 115 to convert the “S” file 508 into a sparse file. A sparse file is a well-known type of file having data within but not filling the file’s logical space (e.g., at the beginning of the file and at the end of the file, and a hole or empty space in between). In converting the “S” file 508 into a sparse file, the portions corresponding to the second file may be zeroed out. These portions are then available for storage of other files or data objects by the operating system on cloud storage sites (e.g., on magnetic disks, but sparse files may be used on other types of cloud storage sites, such as tape or optical disks). Additionally or alternatively, the “S” file may be designated as a sparse file upon its creation.

After 30 days have elapsed, the first and third files are eligible to be pruned. Assuming that there are no intervening storage operations involving files that reference either of these files, both the first and third files can be pruned. The chunk folders 502 corresponding to the first and second storage operations can be deleted, thereby deleting the metadata files 204, the “N” files 506 and the “S” files 508 and recovering the space previously allocated for their storage. (The process for pruning data objects is discussed in greater detail with reference to, e.g., FIGS. 4 and 14.) Therefore, the data structures 500, 510, 520 illustrated in FIGS. 5A through 5C and the techniques described herein also allow for pruning data objects to recover space previously allocated to them on the cloud storage site 115.

Accordingly, the data structures 500, 510, 520 illustrated in FIGS. 5A through 5C and the techniques described herein enable the performance of storage operations cumulatively involving very large numbers of data objects, while still allowing for recovery of space allocated to these data objects when their storage is no longer required. For example, an administrator can back up numerous files across numerous clients and avoid storing redundant copies or instances of the files. The administrator can also easily recover space on the cloud storage site 115 when it is no longer required to store the files, for example, as according to a retention policy that indicates for how long files are to be stored on the cloud storage site 115. Accordingly, the data structures and techniques described herein enable the optimization of storage operations involving very large numbers of data objects.

After having been stored on the cloud storage site 115, files contained in chunk folders may be moved to secondary storage, such as to disk drives, cloud storage sites, or to tapes in tape drives. More details as to these operations may be

US 10,248,657 B2

33

found in the previously referenced U.S. Pat. Pub. No. 2008-0243958, entitled SYSTEM AND METHOD FOR STORING REDUNDANT INFORMATION (U.S. patent application Ser. No. 12/058,367). In moving chunk files to secondary storage, they may be converted into an archive file format. In some examples, the techniques described herein may be used to deduplicate data already stored on secondary storage.

FIG. 5D is an illustration of a data structure 540 for storing chunk folders and their container files in an archive file format. The archive file may be stored on various cloud storage sites, such as on disk drives, magnetic tapes, or cloud storage sites. The archive file includes a chunk 0 542 located at offset 0, a chunk 1 542 located at offset 5, a chunk 2 542 located at offset 10, a chunk 3 542 located at offset 15, and a chunk n located at offset 65. The offsets are in relation to the start of the archive file. More details as to a suitable archive file format may be found in the assignee's U.S. Pat. Pub. No. 2008-0229037, entitled SYSTEMS AND METHODS FOR CREATING COPIES OF DATA, SUCH AS ARCHIVE COPIES (U.S. patent application Ser. No. 11/950,376), the entirety of which is incorporated by reference herein. An archive file may be considered as a container of data objects.

12. Pruning Object-Level Deduplicated Data

Consider the example of a client for which a storage operation job was performed on Jan. 1, 2008, resulting in the creation of an archive file. A retention policy provides that the archive file has to be retained for 30 days. On Jan. 31, 2008, the archive file becomes prunable and thus can be deleted. Deleting the archive file may require deleting data stored in one or more chunks on one or more media. However, the archive file may not be able to be deleted if it is referenced by data objects within other archive files. This is to avoid orphaning data objects, e.g., by deleting a data object when it is still referenced in another archive file. The system keeps tracks of references to data objects in order to avoid orphaning data objects.

To assist in pruning, the deduplication database 299 maintains a primary table and a secondary table. The primary table contains all the single instance records of all data objects for which an identifier was created. For each record in the primary table, the secondary table contains a record that may reference the record in the primary table.

FIGS. 7A and 7B illustrate example primary and secondary tables 700, 750. The primary table 700 has a primary record ID column 710 that may contain primary keys, a file ID column 720 that contains an identifier of a file or data object (e.g., the identifier of the file or data object), and a location column 730 that contains the location of the file or data object (e.g., the archive file ID and its offset within the archive file). The primary table 700 may also contain other columns (not shown).

The secondary table 750 has a secondary record ID column 760 that may contain primary keys, an archive file ID column 765 that contains the archive file ID, a file column 770 that contains the same identifier of the file or data object as in the primary table 700, and a reference_{IN} column 775 that contains an identifier (in the form of an archive file ID and an offset) of a file or data object that references the archive file. The secondary table 750 also has a reference_{OUT} column 780 that contains an identifier (in the form of an archive file ID and an offset) of a referenced file or data object. The secondary table 750 may also contain other columns (not shown).

FIG. 6 is a flow diagram illustrating a process 600 for pruning a deduplication database 299 by pruning or deleting

34

data objects stored in archive files, or entire archive files. As previously noted, archive files can be thought of as containers of data objects. The process 600 begins at step 605 where a selection of an archive file to be pruned is made. This selection can be made manually, such as by an administrator, or automatically, such as by the archive file aging out of a retention policy. At step 610, the media file system agent 240 performs a lookup of the archive file in the primary 700 and secondary tables 700, 750. At step 615, the media file system agent 240 determines if the archive file has references out (e.g., to other archive files).

If the archive file has references out, the process 600 continues to step 620, where the references out are deleted. At step 625, the media file system agent 240 determines if the archive files referenced by the references out have other references in. If there are no other references in, at step 630, the media file system agent 240 prunes the archive files referenced by the references out.

If the archive file does not have any references out (step 615), or if it does, and if the archive files referenced by the references out have other references in (step 625), the process 600 continues at step 635. At this step, the media file system agent 240 determines if the archive file has references in. If it does have references in, this means the archive file cannot be pruned. The process continues at step 640, where the media file system agent 240 deletes the references in. At step 645 the media file system agent 240 adds a reference to the archive file to a deleted archive file table (discussed below).

If the archive file does not have any references in (step 635), the media file system agent 240 prunes the archive file at step 650. The media file system agent 240 then creates an entry in the deleted archive file table for the pruned archive file (if there wasn't already an entry) and adds a deleted timestamp to the entry. If there is already an entry for the pruned archive file, the media file system agent 240 adds a deleted timestamp to the entry at step 655.

FIG. 7C illustrates an example deleted archive file table 752. The deleted archive file table 752 has a primary record ID column 754 that may contain primary keys, an archive file ID column 756 that contains an identifier of the archive file, a reference_{IN} column 758 that contains an identifier (in the form of an archive file ID and an offset) of a file or data object that references the archive file, and a deleted timestamp column 762 that contains a timestamp indicating when the archive file was deleted. In the case of an archive file that has not yet been deleted, the timestamp deleted column would be empty or null in the archive file's entry.

The process 600 will now be explained using the examples of the records shown in the primary and secondary tables 700, 750. At time T₁, the process 600 begins. At step 605, the media file system agent 240 receives a selection of AF₁ to prune. At step 610 the media file system agent 240 looks up AF₁ in the primary and secondary tables 700, 750. At step 615, the media file system agent 240 determines that AF₁ has a reference out, shown by entry 794 in the secondary table 750. (Entry 792 is shown in the secondary table 750 with strikethrough to indicate that it was previously deleted during an operation to prune AF_•.) At step 620, the media file system agent 240 deletes this reference out by deleting entry 794 from the secondary table 750. At step 625, the media file system agent 240 determines if AF_• has any other references in. Since the only reference in for AF_• is from AF₁ (which is to be pruned), AF_• does not have any other references in. At step 630, the media file system agent 240

then prunes AF₁ and adds a timestamp indicating that AF₁ was pruned at time T₁ at entry 772 of the deleted archive file table 752.

At step 635, the media file system agent 240 determines if AF₁ has any references in. AF₁ has a reference in from AF₃, shown in entry 796 of the secondary table 750. The media file system agent 240 thus cannot prune AF₁. At step 640, the media file system agent 240 deletes the references in to AF₁ by deleting entry 796 from the secondary table 750. At step 645, the media file system agent 240 adds entry 774 to the deleted archive file table 752, leaving the deleted timestamp blank. The blank timestamp indicates that AF₁ should be pruned. The process 600 then concludes.

At time T₂, the process 600 begins anew. At step 605, the media file system agent 240 receives a selection of AF₃ to prune. At step 610, the media file system agent 240 looks up AF₃ in the primary and secondary tables 700, 750. At step 615, the media file system agent 240 determines that AF₃ has a reference out, shown by entry 798 in the secondary table 750, which references AF₁. At step 620, the media file system agent 240 deletes entry 798 from the secondary table 750. At step 625, the media file system agent 240 determines if AF₁ has any other references in. Since the only reference in for AF₁ is from AF₃ (which is to be pruned), AF₁ does not have any other references in and can now be pruned. At step 630, the media file system agent 240 then prunes AF₁ and adds a timestamp indicating that AF₁ was pruned at time T₂ at entry 774 of the deleted archive file table 752. This entry now indicates that AF₁ has been pruned at time T₂.

At step 635, the media file system agent 240 determines if AF₃ has any references in. AF₃ has no references in listed in the secondary table 750. The media file system agent thus can prune AF₃. At step 650, the media file system agent 240 prunes AF₃. At step 655, the media file system agent 240 adds the entry 776 to the deleted archive file table 752 with a deleted timestamp as T₂. The process 600 then concludes.

The pruning process 600 thus enables the system to maximize available storage space for storing archive files by storing them efficiently and then deleting or pruning them when it is no longer necessary to store them. The pruning process 600 may have additional or fewer steps than the ones described, or the order may vary other than what is described. For example, instead of the media file system agent 240 adding a timestamp to an entry in the deleted archive file table 752 to indicate when the archive file was pruned, the media file system agent may simply delete the entry from the deleted archive file table 752. As another example, entries in the primary table 700 may also be deleted when the corresponding archive files are deleted. Those of skill in the art will understand that other variations are of course possible.

Sub-Object-Level Deduplication

Instead of deduplication of data objects, deduplication can be performed on a sub-object level in a substantially similar fashion to that described previously with respect to object-level deduplication. A sub-object is a set of blocks that forms a proper subset of all of the blocks within a file or data object. That is, for a file consisting of n blocks, the largest sub-object of the file comprises at most n-1 blocks. An object may thus comprise two or more sub-objects, and be a logical division of the data object. For example, a .pst file may include two or more sub-objects: a first sub-object that stores emails from a user's mailbox, and one or more sub-objects that stores attachments or other data objects associated with the user's mailbox (e.g. subfolders, shared folders, etc.) The deduplication module 299 may include an object division component (not shown) that divides data

objects, such as files, into sub-objects. The object division component may receive files or objects, divide the files into two or more sub-objects, and then deduplicate the two or more sub-objects as described previously with respect to object-level deduplication.

The object division component may perform different processes when determining how to divide a data object. For example, the object division component may include indexing, header, and other identifying information or metadata in a first sub-object and the payload in other sub-objects. The object division component may follow a rules-based process when dividing a data object. The rules may define a minimum or maximum data size for a sub-object, a time of creation for data within a sub-object, a type of data within a sub-object, and so on.

For example, the object division component may divide a user mailbox (such as a .pst file) into a number of sub-objects, based on various rules that assign emails within the mailbox to sub-objects based on the metadata associated with the emails. The object division component may place an index of the mailbox (and its various subfolders) in a first sub-object and all emails for that mailbox in other sub-objects. The object division component may then divide the other sub-objects based on dates of creation, deletion or reception of the emails, size of the emails, sender of the emails, type of emails, and so on. Thus, as an example, the object division component may divide a mailbox as follows:

User1/Sub-object1	Index
User1/Sub-object2	Sent emails
User1/Sub-object3	Received emails
User1/Sub-object4	Deleted emails
User1/Sub-object5	All Attachments.

Of course, other divisions are possible. Sub-objects may not necessarily fall within logical divisions. For example, the object division component may divide a data object based on information or instructions not associated with the data object, such as information about data storage resources, information about a target cloud storage site, historical information about previous divisions, and so on.

Once the division component has divided an object into sub-objects, deduplication of the sub-objects proceeds in substantially the same fashion as described previously with respect to object-level deduplication. To do this, the deduplication module determines, by analyzing data structures in the deduplication database in view of the sub-object's identifier, whether the sub-object of data is already stored on a cloud storage site. If it is, then the secondary storage computing device 1) stores a link to the already stored sub-object of data in a metadata file and 2) discards the sub-object of data from the memory buffer. If it is not already stored, then the secondary storage computing device 165 stores the sub-object of data in a container file. A link or pointer may comprise a URL to a data object or file within a cloud storage site 115A-N.

13. Block-Level Deduplication

Instead of deduplication of files, data objects or sub-objects, deduplication can be performed on a block level. Files can be broken into blocks and deduplicated by the deduplication module 299. Typically blocks are fixed sizes, such as 64 Kb or 128 Kb. In such embodiments, typically, the clients 130 will generate the identifiers, since distributed identifier generation may free up the deduplication module 299 to perform other operations (e.g., storing data, retrieving data, etc.). The clients 130 typically send the blocks of data

US 10,248,657 B2

37

and other data (e.g., metadata and/or the data that is not eligible for deduplication) in a data stream to the deduplication module 299. A deduplication module 299 receives blocks of data from the clients 130 and accesses a deduplication database 297 to determine whether a sufficient number of instances of each block have been appropriately stored. To do this, the system determines, by analyzing data structures in the deduplication database 297 in view of the block's identifier, the number of instances of each block of data that is already appropriately stored on a cloud storage site. During this lookup, to determine whether prior instances were appropriately stored, the system may only consider those instances of the object stored on certain cloud storage sites 115 and/or certain classes of cloud storage sites 115. For example, the deduplication module 299 may restrict the lookup to those cloud storage sites 115 that would satisfy storage policy parameters applicable to each block, such as class of storage used for the object (e.g. data security associated with a particular cloud storage site). Additionally, during this lookup, the deduplication database 297 may restrict the lookup to only those instances of a block stored within a certain time frame. For example, the deduplication database 297 may restrict lookup only to those instances stored within secondary storage in the last seven years.

If an appropriate number of instances of a block have already been appropriately stored, then the deduplication module 299 1) stores a link to the already stored block of data in a metadata file and 2) discards the block of data from the memory buffer. If it is not already stored, the deduplication module 299 stores the block of data in a container file. A link or pointer may comprise a URL to a block or file within a cloud storage site 115A-N.

Because the size of a block of data and associated metadata is typically less than the size of a memory buffer, the deduplication module 299 can keep a single block of data in a single memory buffer while it looks up its identifier in the deduplication database 297. This allows the deduplication module to avoid writing the block of data to a disk (an operation that is typically slower than storing the block of data in a RAM buffer) until the deduplication module determines that it needs to store the block of data in a container file on a cloud storage site. The deduplication module 299 stores data that is not eligible for deduplication in metadata files.

Alternatively, the clients 130 may transmit only the identifiers to the deduplication module 299 for lookup in the deduplication database 297. If the deduplication module 299 determines that an instance of a block has not already been stored on the cloud storage site 115, the deduplication module 299 can instruct the client 130 to send a copy of the block to the deduplication module, which it then stores on the cloud storage site 115. Alternatively, the client 130 itself can send the copy of the block to the cloud storage site 115.

By storing multiple blocks of data in a single container file, the deduplication module 299 avoids storing each block of data as a separate file on the file systems of the cloud storage sites. This reduces the number of files that would be stored on the file systems of the cloud storage sites, thereby ensuring that the cloud storage sites can adequately store the data of the clients 130 in the data storage system.

One advantage of these techniques is that they significantly reduce the number of files stored on a file system of a client or cloud storage site. This is at least partly due to the storage of data blocks within the container files. Even if the deduplication module performs numerous storage operations, these techniques will result in storing far fewer files on the file system than storage operations where each data

38

block is stored as a separate file. Therefore, the file system of the client or cloud storage site may not necessarily have to contend with storing excessively large numbers of files, such as millions of files or more. Accordingly, these techniques enable very large numbers of blocks of data to be stored without regard to limitations of the file system of the client or cloud storage site.

However, the storage of blocks of data in container files may create additional complexities when it comes time to prune or delete data. This is because a container file may contain blocks of data that are referenced by links in metadata files and thus cannot be deleted, as these blocks of data typically still need to be stored on the cloud storage sites. Furthermore, because the blocks of data are not stored as files on the file systems of the cloud storage sites, they cannot be directly referenced by the file system.

The systems and methods described herein provide solutions to these problems. The deduplication module creates the container files as sparse files (typically only on operating systems that support sparse files, e.g., Windows operating systems, but also on other operating systems that support sparse files). A sparse file is type of file that may include empty space (e.g., a sparse file may have real data within it, such as at the beginning of the file and/or at the end of the file, but may also have empty space in it that is not storing actual data, such as a contiguous range of bytes all having a value of zero). Second, the deduplication module maintains a separate index that stores an indication of whether blocks of data in container files are referred to by links in metadata files. In some examples, this can be thought of as creating another file system on top of the existing file systems of the cloud storage sites that keeps track of blocks of data in the container files.

When a block of data is not referred to and does not need to be stored, the deduplication module can prune it. To prune data, the deduplication module accesses the separate index to determine the blocks of data that are not referred to by links. On operating systems that support sparse files, the deduplication module can free up space in the container files corresponding to those blocks of data by marking the portions of the physical media corresponding to the unreferenced portions of the container file as available for storage (e.g., by zeroing out the corresponding bytes in the container files). On operating systems that do not support sparse files, the deduplication module can free up space in the container files by truncating the extreme portions of the container files (e.g., the beginnings and/or the ends of the container files), thereby making the corresponding portions of the physical media available to store other data. Freeing up space in container files allows the operating system to utilize the freed-up space in other fashions (e.g., other programs may utilize the freed-up space).

14. Data Structures for Block-Level Deduplication

FIG. 8 is a diagram illustrating data structures that may be used to store blocks of deduplicated data and non-deduplicated data on the cloud storage site 115 in an archive format. The data structures include one or more volume folders 802, one or more chunk folders 804/805 within a volume folder 802, and multiple files within a chunk folder 804. Each chunk folder 804/805 includes a metadata file 806/807, a metadata index file 808/809, one or more container files 810/811/813, and a container index file 812/814. The metadata file 806/807 stores non-deduplicated data blocks as well as links to deduplicated data blocks stored in container files. The metadata index file 808/809 stores an index to the data in the metadata file 806/807. The container files 810/811/813 store deduplicated data blocks. The container index file

US 10,248,657 B2

39

812/814 stores an index to the container files **810/811/813**. Among other things, the container index file **812/814** stores an indication of whether a corresponding block in a container file **810/811/813** is referred to by a link in a metadata file **806/807**. For example, data block **B2** in the container file **810** is referred to by a link in the metadata file **807** in the chunk folder **805**. Accordingly, the corresponding index entry in the container index file **812** indicates that the data block **B2** in the container file **810** is referred to. As another example, data block **B1** in the container file **811** is referred to by a link in the metadata file **807**, and so the corresponding index entry in the container index file **812** indicates that this data block is referred to.

As an example, the data structures illustrated in FIG. 8 may have been created as a result of two storage operations involving two clients **130**. For example, a first storage operation on a first client **130** could result in the creation of the first chunk folder **804**, and a second storage operation on a second client **130** could result in the creation of the second chunk folder **805**. The container files **810, 811** in the first chunk folder **804** would contain the blocks of deduplicated data of the first client **130**. If the two clients **130** have substantially similar data, the second storage operation on the data of the second client **130** would result in the media file system agent **240** storing primarily links to the data blocks of the first client **130** that are already stored in the container files **810, 811**. Accordingly, while a first storage operation may result in storing nearly all of the data subject to the storage operation, subsequent storage operations involving storage of similar data on the same cloud storage site **115** (or another appropriate cloud storage site) may result in substantial data storage space savings, because links to already stored data blocks can be stored instead of additional instances of data blocks.

If the cloud storage site **115** (or operating system of the cloud storage site) supports sparse files, then when the media file system agent **240** creates container files **810, 811, 813**, it can create them as sparse files. A sparse file is type of file that may include empty space (e.g., a sparse file may have real data within it, such as at the beginning of the file and/or at the end of the file, but may also have empty space in it that is not storing actual data, such as a contiguous range of bytes all having a value of zero). Having the container files **810, 811, 813** be sparse files allows the media file system agent **240** to free up space in the container files **810, 811, 813** when blocks of data in the container files **810, 811, 813** no longer need to be stored on the cloud storage sites **115**. In some examples, the media file system agent **240** creates a new container file **810, 811, 813** when a container file either includes 100 blocks of data or when the size of the container file **810** exceeds 50 Mb. In other examples, the media file system agent **240** creates a new container file **810, 811, 813** when a container file satisfies other criteria (e.g., it contains from approximately 100 to approximately 1,000 blocks or when its size exceeds approximately 50 Mb to 1 Gb). Those of skill in the art will understand that the media file system agent **240** can create a new container file **810, 811, 813** when other criteria are met.

One advantage of the data structures illustrated in FIG. 8 and/or of the techniques described herein is that they significantly reduce the number of files transferred and stored on a file system of the cloud storage site **115**. This is at least partly due to the storage of data blocks within the container files **810, 811, 813**. Even if numerous storage operations using these data structures are performed, there will be far fewer files on the cloud storage site **115** than there would be in storage operations where each data block is stored as a

40

separate file. Therefore, the client computers need not transfer certain blocks or files, and the file system of the cloud storage site **115** may not necessarily have to contend with storing excessively large numbers of files, such as millions of files or more. Accordingly, the systems and methods described herein enable very large numbers of blocks of data to be stored without regard to limitations of the file system of the cloud storage site **115**.

Another advantage is that the data storage system enables a reduction in the amount of blocks of data stored on the cloud storage sites **115**, while still maintaining at least one instance of each block of data in primary data. In examples where the data storage system stores a variable number of instances of blocks of data, blocks of data can be distributed across two or more cloud storage sites **115**, thereby adding a further aspect of redundancy.

Another advantage is that the metadata files **806, 807**, the metadata index files **808, 809**, the container files **810, 811, 813**, and/or the container index files **812, 814** could be used to replicate the data stored in the deduplication database **297**, or to reconstruct the deduplication database **297** if the data of the deduplication database **297** is ever lost and/or corrupted.

The storage of data blocks in the container files may create additional complexities when it comes time to prune (delete) data blocks that the data storage system no longer need retain. This is because the data blocks are not stored as files on the file system on the cloud storage site **115** and thus cannot be directly referenced by the file system. As described in detail herein, the media file system agent **240** uses the container index files **812, 814** to keep track of which blocks of data are referenced and thus which blocks are not prunable (deletable).

In some examples, the use of the container index files **812, 814**, the metadata index files **808, 809**, and/or the primary and secondary tables **700, 750** to track data acts as a driver, agent or an additional file system that is layered on top of the existing file system of the cloud storage site **115**. This driver/agent/additional file system allows the data storage system to efficiently keep track of very large numbers of blocks of data, without regard to any limitations of the file systems of the cloud storage sites **115**. Accordingly, the data storage system can store very large numbers of blocks of data.

Accordingly, the data structures illustrated in FIG. 8 and the techniques described herein enable the performance of multiple storage operations cumulatively involving very large amounts of data, while still allowing for recovery of space on the cloud storage site **115** when storage of certain data blocks is no longer required. For example, the data of numerous clients **130** can be protected without having to store redundant copies or instances of data blocks. Space on the cloud storage site **115** can also be recovered when it is no longer necessary to store certain data blocks. Accordingly, storage operations involving very large amounts of data are enabled and optimized by the techniques described herein.

15. Deduplication Databases to Enable Containerized Deduplication to Cloud-Based Storage

In some embodiments, the deduplication database **297** may maintain a primary block table and a secondary block table. The primary table may include an identifier column in which a data block identifier is stored, a location column in which a location of the data block in a container file is stored, an offset column indicating the offset within the container file corresponding to the location of the data block, and a reference count column, which contains a reference

US 10,248,657 B2

41

count of the number of links that refer to the data block. The location column may include URLs that indicate storage locations on cloud storage sites 115A-N. An example primary block table is shown below in Table 1.

TABLE 1

Primary Block Table			
Identifier	Location	Offset	Reference Count
0xA1B3FG	http://www.storecloud.com/companyname/V_3/Chunk_1/Container File 001	10	2
0xFG329A	http://www.storecloud.com/companyname/V_1/Chunk_5/Container File 002	6	0
0xC13804	http://www.storecloud.com/companyname/V_2/Chunk_1/Container File 001	38	1
...

For example, row 1 includes information about a data block for which the identifier is "0xA1B3FG." This data block is located in the container file that is indicated in the location column, at an offset of 10 within the container file. As shown, the URL indicates a cloud storage site ("storecloud.com") used to store the container file. As indicated in the reference count column, this data block is referred to twice, meaning that there are two links that refer to the data block. As another example, row 2 includes information about a data block for which the identifier is "0xC13804." The location of this data block is indicated in the location column at an offset of 38 within the container file, and it is referred to one other time, by one link.

A secondary block table includes information about links that refer to data blocks. The secondary block table includes an identifier column, a referring location column, and an offset column. The referring location column may include URLs that indicate storage locations on cloud storage sites 115A-N. An example secondary block table is shown below in Table 2.

TABLE 2

Secondary Block Table		
Identifier	Referring Location	Offset
0xA1B3FG	http://www.storecloud.com/companyname/V_3/Chunk_1/MetaDataFile 001	5
0xA1B3FG	http://www.2ndCloud.com/co_name/V_4/Chunk_18/MetaDataFile 003	15
0xC13804	http://www.storecloud.com/companyname/V_3/Chunk_2/MetaDataFile 001	19

For example, the first row includes information about a reference to the data block having the identifier of "0xA1B3FG" (the first row in the primary block table). The location of the link (within a first cloud storage site) is indicated in the second column, at an offset of five within the indicated metadata file. As another example, the second row includes information about another reference to the data block having the identifier of "0xA1B3FG." This location of the link (within a second cloud storage site "2ndCloud") is indicated in the second column, at an offset of 15 within the indicated metadata file. As another example, the third row includes information about a reference to the block for which the identifier is "0xC13804" (the second row in the primary block table). The location of the link is indicated in the second column, at an offset of 19 within the indicated metadata file.

42

The system may maintain similar primary and secondary tables to facilitate object-level and/or sub-object level deduplication processes. For example, a deduplication database 297 may maintain a primary object table and a secondary object table having similar fields to those shown in Tables 1 and 2, respectively. In such an example, each entry in a primary object table corresponds to a stored data object. Each entry in a primary object table corresponds to a reference to a stored data object.

16. Pruning Block-Level Deduplicated Data

FIG. 9 is a flow diagram of another process 900 for pruning deduplicated data blocks that may be employed in some examples. The process 900 is described as being performed by the media file system agent 240, although those of skill in the art will understand that aspects of the process 900 may be performed by any of the entities described herein. The process 900 begins at step 905 when the media file system agent 240 receives instructions to prune data corresponding to a storage operation (job). Additionally or alternatively, one or more files can be selected to be pruned, and/or one or more data blocks can be selected to be pruned. This selection of a job or other data to be deleted can be made manually, such as by an administrator, or automatically, such as by the job, files, and/or data blocks aging out by a retention policy.

As previously noted, the data structures illustrated in FIG. 8 may have been created as a result of two jobs involving two clients 130. For example, a first job on a first client 130 could result in the creation of the first chunk folder 804, and a second job on a second client 130 could result in the creation of the second chunk folder 805. The process 900 is described using this example. More specifically, the process 900 is described below as pruning the data created as a result of the first job. Of course, a similar process may be used to delete other jobs, or even smaller increments of data or data objects, such as individual files or blocks.

At step 907 the media file system agent 240 determines the file, e.g., archive file, and the volume folders 802 and chunk folder 804 corresponding to the job to be pruned. The media file system agent 240 may do so, for example, by analyzing various data structures to determine this information. At step 910 the media file system agent 240 deletes the metadata file 806 and the metadata index file 808 in the chunk folder 804. The media file system agent 240 can delete the metadata file 806 and the metadata index file 808 in this example because these files include data that is not referenced by any other data.

At step 915 the media file system agent 240 accesses the container file 810 and the container index file 812 in the chunk folder 804. The media file system agent 240 begins iterating through the data blocks in the container files 810. At step 920, beginning with a first block in the container file 810, the media file system agent 240 accesses the primary block table in the deduplication database 297. The media file system agent 240 determines from the primary block table whether the reference count of a data block in the container file 810 is equal to zero. If so, this indicates that there are no references to the data block. The process 900 then continues at step 925, where the media file system agent 240 sets the entry in the container index file 812 corresponding to the data block equal to zero, thus indicating that there are no references to the data block, and it is therefore prunable.

If the reference count of a data block is not equal to zero, then the data block is not prunable, and the process 900 continues at step 930. At this step, the media file system agent 240 determines whether there are more data blocks in the container file 810. If so, the process 900 returns to step

US 10,248,657 B2

43

920, where it accesses the next data block. If there are no more data blocks in the container file **810**, the process **900** continues at step **932**, where the media file system agent **240** determines whether all the entries in the container index file **812** corresponding to the container file **810** are equal to zero. As illustrated in FIG. **8**, the second index entry in the container index file **812** is not equal to zero, thus indicating that the corresponding block in container file **810** is referenced (by data in the chunk folder **805**, as earlier described). Accordingly, the container file **810** cannot be deleted.

However, if the container file **810** did not contain any referenced data blocks, then at step **933**, the media file system agent **240** would delete the container file **810**. The process would then continue at step **935**, where the media file system agent **240** determines whether there are more container files. According to the example as illustrated in FIG. **8**, there is an additional container file **811**. The process **900** then returns to step **915**, where it performs the same steps **920-933** for container file **811**. As a result of performing these steps, the media file system agent **240** would also determine that the container file **811** cannot be deleted, because it contains a data block that is referenced (by data in the chunk folder **805**, as earlier described).

After processing container files **810**, **811**, the process **900** continues at step **940**, where the media file system agent **240** determines whether to free up storage space in the container files **810**, **811**. The media file system agent **240** may do so using various techniques. For example, if the operating system of the media file system agent **240** supports sparse files, then the media file system agent **240** may free up space by zeroing out the bytes in the container files corresponding to the space to be freed up. For a number of contiguous blocks (e.g., a threshold number of contiguous blocks, such as three contiguous blocks) for which the corresponding entries in the container index file **812** indicate that the blocks are not being referred to, then the media file system agent **240** may mark these portions of the container files **810**, **811** as available for storage by the operating system or the file system. The media file system agent **240** may do so by calling an API of the operating system to mark the unreferenced portions of the container files **810**, **811** as available for storage.

The media file system agent **240** may use certain optimizations to manage the number of times portions of the container file are marked as available for storage, such as only zeroing out bytes in container files when a threshold number of unreferenced contiguous blocks is reached (e.g., three unreferenced contiguous blocks). These optimizations may result in less overhead for the operating system because it reduces the number of contiguous ranges of zero-value bytes in the container files **810**, **811** that the operating system must keep track of (e.g., it reduces the amount of metadata about portions of the container files **810**, **811** that are available for storage).

If the operating system of the media file system agent **240** does not support sparse files, then the media file system agent **240** may free up space by truncating either the beginning or the end of the container files **810**, **811** (removing or deleting data at the beginning or end of the container files **810**, **811**). The media file system agent **240** may do so by calling an API of the operating system, or by operating directly on the container files **810**, **811**. For example, if a certain number of the last blocks of the container file are not being referred to, the media file system agent **240** may truncate these portions of the container files **810**, **811**. Other techniques may be used to free up space in the container files **810**, **811** for storage of other data. At step **945** the media file

44

system agent **240** frees up space in the container files **810**, **811**. The process **900** then concludes.

As a result of the process **900**, the chunk folder **804** would contain only the container files **810**, **811** and the container index file **812**. At a later time, when the chunk folder **805** is pruned (when the job that created this chunk folder is selected to be pruned), then the container files **810**, **811** in the chunk folder **804** can be deleted, because they no longer contain data blocks that are referenced by other data. Therefore, pruning data corresponding to a job may also result in pruning data corresponding to an earlier job, because the data corresponding to the earlier job is no longer referenced by the later job.

Although the process **900** is described with reference to the pruning of data corresponding to jobs (one or more storage operations), other data can also be pruned. For example, an administrator may wish to delete deduplicated data but retain non-deduplicated data. In such case, the administrator may instruct the media file system agent **240** to delete the container files **810**, **811**, **813** but retain the metadata files **806**, **807** and metadata index files **808**, **809**. As another example, an administrator or storage policy may delete one or more specific files. In such case, the media file system agent **240** deletes the data blocks in the container files **810**, **811**, **813** corresponding to the specific files but retains other data blocks. The process **900** may include fewer or more steps than those described herein to accommodate these other pruning examples. Those of skill in the art will understand that data can be pruned in various fashions and, therefore, that the process **900** is not limited to the steps described herein.

17. Containerizing Deduplicated Data for Storage in the Cloud

During a storage operation that utilizes deduplication, it may be desirable to determine a suitable container file size, particularly if the storage operation will result in the container files being stored on a target cloud storage site **115A-N**. As described previously, a single storage operation that utilizes deduplication may result in as few as three container files being created in a secondary cloud storage site **115**, such as three for each company storing data to that cloud storage site. The contents of the few container files may reflect the content of thousands of data objects and/or millions of data blocks in primary storage. By containerizing the objects or blocks, the system reduces the strain on the file system namespace of the secondary cloud storage site **115**, since it reduces the number of files stored on the file system of the cloud storage site **115**. The fewer container files used per storage operation, the less strain there is on the file system namespace of the secondary cloud storage site **115**. Thus, by using larger container files, the system may reduce namespace strain on the secondary cloud storage site **115**.

When creating or writing container files to a target cloud storage site **115A-N** used as a secondary cloud storage site, the characteristics of the WAN network connection used to transfer the container files from the media file system agent **140** to the cloud storage site **115A-N** may impose other restrictions upon the size of container files used. For example, the bandwidth of the network connection may impose an upper limit on the size of container files that may be used (e.g., an upper limit of approximately 1000 blocks). If the network connection has low bandwidth, the upload of large container files to the cloud storage site may prove prohibitively slow. Also, the restoration of a particular data object or block may require the retrieval of the entire container file comprising that data object/block from the cloud storage site; if the container file is too large for a

US 10,248,657 B2

45

low-bandwidth network, then restoration times may become prohibitively slow. As another example, the latency of the network connection may impose a lower limit on the size of container files that may be used. This is because the total time needed to perform a storage operation may be increased if for each container file created and transferred to the target cloud storage site, the system must slowly transmit the container file and/or await a response from the cloud storage site **115A-N** before processing the next container file in the storage operation.

Other factors may also affect the choice of size for container files. For example, some cloud storage sites **115A-N** may not support sparse files and thus not support sparsification of container files. In this situation, smaller container files may be desirable, because then it becomes more likely the system will be able to prune entire container files from the cloud storage site **115A-N**, even if it cannot prune out individual blocks/objects using sparsification techniques. As another example, a particular cloud storage site **115A-N** may have a pricing structure that charges both for the total amount of storage used (e.g., total gigabytes or petabytes used) and the number of files or directories used on the site. If the cloud storage site **115A-N** bases its charges on the number of files or directories used on the site, larger container files may be desirable. In some embodiments, the system may also additionally impose an absolute upper or lower limit on the size of container files used. For example, the system may impose an upper limit on the size of container files in order to minimize the amount of time it takes the system to traverse a container file during data restoration. For example, in some embodiments, the system may impose an absolute 100 block size upon container files, even if the network bandwidth would theoretically allow for larger container files. As another example, the system may impose an absolute lower limit on the size of container files used, since there may be overhead costs (e.g., processing time and/or memory used) for each additional container file used in a storage operation.

Thus, the deduplication module **299** or another system component may perform the following process to establish a container size for a storage operation. The deduplication module **299** or system may (1) determine the average latency and bandwidth of the network connection between the target cloud storage site **115A-N** and the media file system agent **240** (or similar metrics regarding the network connection, e.g., maximum latency and minimum bandwidth), (2) determine any namespace restrictions imposed by the target cloud storage site **115A**, (3) determine whether the target cloud storage site **115A-N** supports the sparsification of data files, (4) determine the pricing structure used by the target cloud storage site, (5) determine any caps set by the system upon container file size, and (6) perform an optimization to establish a container size for the storage operation reflecting one or more of these determined factors and/or other factors (e.g., such as user input).

Alternatively, the system may permit a user to select the container size that will be used for one or more storage operations. Still alternatively, the user or the system may establish for all storage operations, the container size that will be used for a particular cloud storage site or all cloud storage sites.

18. Indexing of Data

As noted above for FIG. **3B**, the system may index data to be stored at a cloud storage site, such as before the data is sent to the cloud storage site. Some details on suitable content indexing techniques will now be presented. Further details may be found in the assignee's U.S. Patent Publica-

46

tion No. 2009-0287665, filed Jul. 29, 2009, entitled METHOD AND SYSTEM FOR SEARCHING STORED DATA (U.S. patent application Ser. No. 12/511,653). FIG. **10** is a flow diagram that illustrates the processing of a content indexing component **205** for later searching, according to one embodiment. The component is invoked when new content is available or additional content is ready to be added to the content index. In step **1010**, the component selects a copy of the data to be indexed. For example, the copy may be a secondary copy of the data, a data snapshot, or data stored or being stored in an archive copy. In step **1020**, the component identifies content within the copy of the data. For example, the component may identify data files such as word processing documents, spreadsheets, and presentation slides within the secondary data store. The system may check the data against previously indexed data, and only index new or additional data. In step **1030**, the component updates an index of the content to make the identified content available for searching. The system may parse, process, and store the data. For example, the component may add information such as the location of the content, keywords found within the content, and other supplemental information about the content that may be helpful for locating the content during a search. In one example, the content indexing component updates a content index stored within the SS index **261**, SS light index **247** and/or the management light index **245** and/or management index **211**. After step **1030**, these steps conclude.

FIG. **11** illustrates some of the data structures used by the system to facilitate content indexing. While the term "field" and "record" are used herein when describing certain data structures, the system described herein may employ any type of data structure. For example, relevant data can have preceding headers, or other overhead data preceding (or following) the relevant data. Alternatively, relevant data can avoid the use of any overhead data, such as headers, and simply be recognized by a certain byte or series of bytes within a serial data stream. Any number of data structures and types can be employed herein.

FIG. **11** illustrates a data structure containing entries of a content index. In some embodiments, a copy of the content index shown (or a copy of a subset of the content index shown) may be stored within the SS index **261**, SS light index **247** and/or the management light index **245** and/or management index **211**. The offline content indexing system uses this and similar data structures to provide more intelligent content indexing. For example, the offline content indexing system may index multiple copies of data and data available from the multiple copies using a secondary copy of data stored on media with a higher availability based on the location or other attributes indicated by the data structure described below. As another example, the offline content indexing system may prefer an unencrypted copy of the data to an encrypted copy to avoid wasting time unnecessarily decrypting the data.

The table **1100** contains a location column **1110**, a keywords column **1120**, a user tags column **1130**, an application column **1140**, and an available column **1150**. The table **1100** contains five sample entries. The first entry **1160** specifies that the location of a file is on a corporate intranet by using a web universal resource locator ("URL"). The entry **1160** contains keywords "finance," "profit," and "loss" that identify content within the file. The entry **1160** contains tags added by a user that specify that the content comes from the accounting department and is confidential. The entry **1160** indicates that a spreadsheet program typically consumes the content, and that the entry is immediately available.

US 10,248,657 B2

47

Another entry **1170** specifies that data is stored on a local tape that is a personal email, and can be available in about an hour. Another entry **1180** specifies an offsite tape holds a presentation related to a cancelled project. The entry **1180** refers to offsite data that is available within one week due to the delay of retrieving the archived data from the offsite location. Another entry **1190** specifies that the location of a word processing document containing data relating to CEO compensation is in a cloud storage site by using a URL that points to a deduplicated archive file that may be implemented by a data structure similar to those shown in FIGS. **5A-D** and/or FIG. **8**. As shown, the estimated retrieval time from this cloud storage site is 15 minutes. Another entry **1195** specifies that the location of a personal email relating to a medical condition is stored in a second cloud storage site by using another URL that points to a deduplicated archive file that may be implemented by a data structure similar to those shown in FIGS. **5A-D** and/or FIG. **8**. As shown, the estimated retrieval time from this cloud storage site is 1 hour.

19. Policy-Driven Storage of Data Across Cloud Storage Sites

Referring again to FIG. **3B**, at step **330**, the system stores deduplicated data (or “dehydrated data”) in secondary cloud storage by utilizing the media file system agent **240** to perform file system operations (such as a “write” operation) on a target cloud storage site **115A**. To determine which target cloud storage site the media file system agent **240** should write to, the media file system agent **240** may retrieve an applicable storage policy (described previously with respect to FIG. **2**) and act in accordance therewith. For example, the media file system agent **240** may retrieve a storage policy stored in management index **211** that specifies that all email objects (and blocks contained therein) should be stored on cloud storage site **115A**, while document objects (and blocks contained therein) should be stored on cloud storage site **115B**. As another example, the storage policy stored in management index **211** may specify that all objects related to a particular client **130** or particular user (e.g., a company CEO) should be stored on a more expensive or reliable cloud storage site **115A** while all other objects for all other clients **130** and/or users should be stored on a less expensive or less reliable cloud storage site **115B**. As yet another example, at block **330**, the system may review the historical performance achieved by various target cloud storage sites **115A-N** to determine which sites have historically achieved the desired performance metrics mandated by a storage policy. Additionally, the system may select a cloud storage site that has better historical performance than other sites.

As another example, a storage policy may specify that a first type of files should be retained for one year in a first target cloud storage site **115A**, that a second type of files should be retained for seven years in a second cloud storage site **115B**, and that a third type of files should be retained indefinitely in a third cloud storage site **115N**. As yet another example, a storage policy may specify that a first type of files (e.g., secondary disk copies needed for rapid disaster recovery) be stored only in storage sites **115**, including cloud storage sites **115A-N**, that can provide sufficient bandwidth, network capacity or other performance to ensure that the time needed to recover a file from the storage device **115** (e.g., cloud storage site **115A-N**) is less a specified recovery time objective.

48

20. Restoring Dehydrated Data Objects from Cloud Storage Sites

After a storage operation has resulted in the storage of dehydrated data on a cloud storage site **115A-N**, it may be necessary to later restore some or all of the original data files, objects, sub-objects, or blocks that were archived during the storage operation. For example, a user or customer of a cloud storage site may wish to retrieve a file that was copied to the cloud storage site in dehydrated form if a primary copy of that file is no longer available on the user's client **130**. As another example, to comply with an electronic discovery request, it may be necessary to retrieve an archived version of a particular file. Some details on suitable techniques for restoring files and objects from dehydrated data will now be presented. Further details may be found in the assignee's U.S. patent application Ser. No. 12/565,576, filed Sep. 23, 2009, entitled **SYSTEMS AND METHODS FOR MANAGING SINGLE INSTANCING DATA**.

FIG. **12** is a flow diagram illustrating a process **1200** for restoring or retrieving data from chunk folders in an archive file format on secondary storage. This process may be utilized to restore data objects stored on cloud storage sites **115A-N**. In order to do so, the system identifies the cloud storage site **115**, the archive file on that cloud storage site, the chunk file within that archive file, and further the location of the data object within that chunk file. At step **1205** a selection of a data object to restore is received, such as from an administrator via a graphical interface. The process of restoring data that has been deduplicated may be referred to herein as “rehydrating deduplicated data.”

At step **1210** the media file system agent **240** is consulted to determine an archive file ID and an offset of the data object to be restored. The media file system agent **240** can determine this information from a data structure, such as a tree index (for example, a c-tree may be used, which, in some examples, is a type of self-balancing b-tree), that it maintains for each archive file. For example, an archive file may be based on files **1** through **n**, with file **1** at offset **1**, file **2** at offset **2**, file **n** at offset **n**, and so on. The media file system agent **240** maintains one tree index per full storage operation cycle. (A storage operation cycle consists of a cycle from one full storage operation of a set of data, including any intervening incremental storage operations, until another full storage operation is performed.) FIG. **13A** illustrates an example data structure **1300** that the media file system agent **240** maintains. The data structure **1300** includes an archive file ID item **1310** that contains the identifier of archive files, a file or data object item **1320** that contains the identifier of the file or data object, and an offset **1330** containing the offset of the file or data object within the archive file or cloud container.

The media file system agent **240** may also maintain a multiple-part identifier, such as a five-part identifier, that includes an enterprise or domain identifier (e.g., an identifier of a company/customer, a grouping of clients/companies, etc.), a client identifier to identify a particular company, customer or host computer to connect to at the customer, an application type (e.g. if all Microsoft Word documents are stored together), a storage operation set identifier to identify when the storage operation data was obtained, and a sub-client identifier to provide a further level of granularity within an enterprise to identify an origin, location, or the use of the data (e.g., a file system on a client could be a sub-client, or a database on a client could be a sub-client).

Using the data structure maintained for the archive file, the media file system agent **240** determines the archive file ID and offset within the archive file of the data object to be

US 10,248,657 B2

49

restored. The media file system agent **240** then needs to determine which chunk contains the data object. To do so, the media file system agent **240** consults another server, such as a storage manager **105** (discussed below), that has a data structure that maps the archive file ID and offset to the specific media (as well as the specific chunk file within the specific media, optionally). For example, the storage manager may maintain a database table that maps the archive file ID to specific media, to a URL indicating the cloud storage site location, or to a bar code number for a magnetic tape cartridge storing that archive file.

FIG. 13B illustrates an example data structure **1350** that the storage manager **109** maintains. The data structure **1350** includes an archive file ID item **1370** identifying a client, a storage operation job, a cycle, and an archive file ID, a media chunk item **1380** containing an identification of the media containing the archive file and the chunk on the media that contains the archive file, and a start item **1390** that contains the archive file ID, an offset, and a size. When utilizing a cloud storage site, some or all of the entries in the media chunk column **1380** may comprise a URL (e.g., a URL like https://www.cloudstorage.com/companyname/CJ/Y/1/C_1.xml) that reflects the location of the archive file within a specific cloud storage site and/or reflects a website where the system may otherwise access the archive file. The media file system agent **240** then can consult a deduplication database **297** to determine the specific chunk that corresponds to the data object to be restored.

At step **1215**, the cloud storage server accesses a particular secondary storage device and the specific media, such as a specific folder within a disk at a cloud storage site (indicated by a URL) or a specific tape cartridge in an automated tape library, is accessed. At step **1220** the cloud storage server opens the specific chunk folder, and the metadata file is accessed. At step **1225**, the metadata file is parsed until the stream header corresponding to the data object or block to be restored is accessed. At step **1230**, the cloud storage server determines the location of the file from the stream data. The stream data indicates the location of the data object to be restored, which is either in a container file in the chunk folder or within a container file in another chunk folder. At step **1235** the data object is retrieved or opened, and the data object is read and streamed back to restore it for the requesting client/host/customer (block **1240**). Each data object may have a piece of data appended to it (e.g., an EOF marker) that indicates to the reader when to stop reading the data object. A similar piece of data may be prepended (e.g., a BOF marker) to the data object. The process **1200** then concludes.

Although the process of FIG. 12 and the data structures of FIG. 13 were described with respect to object-level restoration and retrieval, one having skill in the art will appreciate that a system may employ a similar process and similar data structures to restore and retrieve individual blocks or sub-objects archived within a system.

21. Local Searching of Data Stored on Remote Cloud Storage Sites

As described previously, during the process of FIG. 3B, the system may generate one or more copies of a content index as shown in FIG. 11 within the SS index **261**, SS light index **147**, the management light index **245** and/or management index **211**. Using this content index information, the system may provide local search capabilities. Some details on suitable searching techniques will now be presented. Further details may be found in the assignee's U.S. Patent Publication No. 2008-0091655, filed Mar. 30, 2007, entitled METHOD AND SYSTEM FOR OFFLINE INDEXING OF

50

CONTENT AND CLASSIFYING STORED DATA (U.S. patent application Ser. No. 11/694,869). For example, the storage manager **105** may receive and process a request to search the management index **211** for files matching certain search criteria, and then return matching files. By providing local searching of the content index information, the system may provide more cost-effective and/or faster searches of data archived or stored on a remote cloud storage site **115A-N**, since local searches of a local content index typically do not require file system calls to a cloud storage site other than to retrieve identified files stored therein.

FIG. 14 is a flow diagram that illustrates the processing of a search request by the system, in one embodiment. In step **1410**, the system receives a search request specifying criteria for finding matching target content. For example, the search request may specify one or more keywords that will be found in matching documents. The search request may also specify boolean operators, regular expressions, and other common search specifications to identify relationships and precedence between terms within the search query. In step **1420**, the system searches the content index to identify matching content items that are added to a set of search results. For example, the system may identify documents containing specified keywords or other criteria and add these to a list of search results. In step **1425**, the system generates search results based on the content identified in the content index. In step **1430**, the system selects the first search result. In decision step **1440**, if the search result indicates that the identified content is archived, then the system continues at step **1450**, else the system continues at step **1455**. For example, the content may be archived because it is on a remote cloud storage site.

In step **1450**, the system retrieves the archived content, which may utilize the data restoration methods discussed herein. Additionally or alternatively, the system may provide an estimate of the time required to retrieve the archived content and add this information to the selected search result. In decision step **1455**, if there are more search results, then the system loops to step **1430** to get the next search results, else the system continues at step **1460**. In step **1460**, the system provides the search results in response to the search query. For example, the user may receive the search results through a web page that lists the search results, or the search results may be provided to another system for additional processing through an API. The system may also perform additional processing of the search results before presenting the search results to the user. For example, the system may order the search results, rank them by retrieval time, and so forth. After step **1460**, these steps conclude.

22. Collaborative Searching

In some implementations, a cloud storage site may be integrated with a collaborative search system and collaborative document management system to facilitate collaborative searching, data retrieval, and discovery. Some details on collaborative searching are provided below; further details may be found in the assignee's U.S. Patent Publication No. US-2008-0222108-A1, filed Oct. 17, 2007, entitled METHOD AND SYSTEM FOR COLLABORATIVE SEARCHING (U.S. patent application Ser. No. 11/874,122). Referring to FIG. 25, a block diagram **2500** illustrating an architecture for integrating a collaborative search system with a collaborative document management system is shown. A browser **2505** is used by collaborative participants as an interface to access the integrated system. A collaborative participant submits queries, receives results, and performs other collaborative tasks through the browser **2505**. The browser **2505** is connected to a collaborative document

US 10,248,657 B2

51

management system **2510**, such as the Microsoft SharePoint Server. The collaborative document management system **2510** provides a web-based portal for collaboration between collaborative participants. The collaborative document management system **2510** is connected to a collaborative search system **2520**. The collaborative search system **2520** integrates with the collaborative document management system **2510** and adds additional components, such as web components and content parsers, and provides access to cloud storage content. The collaborative search system **2520** is connected to not only one or more cloud storage sites **115**, but also to local storage (e.g. a storage operation cell **150**), as well as to a security system **2540**, and a document retention system **2550**.

The storage operation cell **150**, as shown in FIG. 2, provides fast access to content from various computer systems within an enterprise. The security system **2540** provides users and groups that are meaningful to a particular enterprise to facilitate searching. The security system **2540** also enforces access rights to collaborative content. The document retention system **2550** places a legal hold on documents related to a document retention request.

In some examples, the collaborative search system receives criteria for a search through a collaborative process. For example, one collaborative participant may create a new query for responding to a discovery request regarding a product made by the company that employs the collaborative participant. The first collaborative participant may add search criteria including the product name and then submit the search criteria to the collaborative document management system **2510** as a collaborative document. Another collaborative participant may open the collaborative document and add additional search criteria, such as instructions to narrow the list of departments from which documents should be searched. For example, the second participant may include the engineering, marketing, and sales teams that worked on the product. The collaborative search system **2520** may also add additional criteria inferred from the criteria added by the collaborative participants. For example, based on the company's indexed data the collaborative search system may determine that two employees, one in a department already within the search criteria and another outside of the current search criteria, frequently send email about projects. Based on this information, the collaborative search system may add the user that is outside of the current search criteria to the search criteria, or it may prompt one of the collaborative participants to consider adding the user to the search criteria.

Alternatively or additionally, the system may provide further features. For example, the system may add additional search criteria inferred from dynamic changes made to the search criteria. The system may use heuristics type information when determining search criteria. The collaborative search system **2520** may define workflows that define the set of steps that are part of completing a task. The collaborative search system **2520** may create a collaborative document based on a set of search results. The collaborative document provides a mechanism for multiple collaborative participants to contribute to steps within a workflow subsequent to the search process. In the example of a discovery request, the steps of performing various levels of review of found documents can consume the majority of the time spent responding to the discovery request, and a collaborative participant may reviewing each document and flagging the document if it contains privileged content or directly add comments to documents within the search results. The collaborative search system **2520** provides a user interface

52

through which a collaborative participant may select from a set of templates that define common search tasks, such as a Sarbanes-Oxley template that initiates a search for materials required to be disclosed under the Sarbanes-Oxley Act.

The user interface of the collaborative search system **2520** may include custom-developed web components to assist with the integration with the collaborative document management system. For example, Microsoft SharePoint Server provides an object model and API for accessing collaborative features such as workflows and a search front-end that can be invoked from custom web pages using the Active Server Page Framework ("ASPX"). The collaborative search system **2520** provides a user interface that does not require specialized software to be installed on the searching client system. The collaborative search system may also provide a set of parsers for viewing content from many different sources, such as received in a list of search results, as web content. For example, the collaborative search system may provide a parser for converting a word processing document into a Hypertext Markup Language ("HTML") web page. Other parsers may convert spreadsheet content, database tables, instant messaging conversation logs, email, or other structured or unstructured content into a web page format accessible via a collaborative participant's browser. In this way, heterogeneous data from many different applications is available through a unified search user interface.

FIG. 26 illustrates the integration of parsers with the collaborative document management system. The collaborative document management system **2510** contains a configuration database **2630**, a schema file **2640**, one or more dynamic web pages **2620**, and one or more generated web pages **2610**. When a collaborative participant accesses the collaborative document management system **2510**, the collaborative document management system **2510** consults the configuration database to determine what to display to the collaborative participant based on factors such as the identity of the user, the particular web address the collaborative participant requested, the access rights of the collaborative participant, the state of previous requests by the collaborative participant to the collaborative document management system, and so on. Based on the determined information to display, the collaborative document management system consults the schema file **2640** to determine the layout of the information for display to the collaborative participant. The schema file **2640** may include instructions based on predetermined layouts, dynamically determined layouts, templates to be included in the layout, and so on. At this point, one or more parsers **2650** may be consulted to migrate data from one or more document types (e.g., **2660** and **2670**) to an XML or other common format. The schema data is passed to an ASPX or other dynamic page **2620** that may use scripts and an object model provided by the collaborative document management system to identify, parse data types, and dynamically build a page with the content that will be displayed to the collaborative participant. For example, the system may present one or more templates described above. After the scripts are run, the dynamic page **2620** generates an HTML or other generic formatted page **2610** that is sent to the collaborative participant's browser/GUI that will be displayed to the collaborative participant.

The collaborative search system **2520** may integrate components for searching data from multiple operating systems and multiple data formats from multiple cloud storage sites. For example, file system data on a Microsoft Windows computer system may be stored differently from file system data on a Linux computer system, but the collaborative search system may make both types of file system data

US 10,248,657 B2

53

available for searching. Data may be gathered from each of these types of disparate data sources and forwarded to a uniform database where the data can be collected, tagged with various classifications, and indexed for searching. The system may then display the data on differently formatted browsers.

Other implementations may integrate a collaborative document management system **2510** and collaborative search system **2520** with another type of storage system that provides content indexing and search capabilities comparable to the storage operation cell **150** shown FIG. **2**. For example, an implementation may integrate a collaborative document management system and collaborative search system with a system shown in FIG. **15**, FIG. **21** and/or FIG. **22**, which are described in greater detail herein.

In some examples, the collaborative search system **2520** integrates information from the security system **2540**. For example, the collaborative search system may use Microsoft Windows Active Directory to determine users whose content should be searched as part of a discovery request. Active Directory contains all of the users in an organization and organizes the users into groups. The security system may provide restrictions on access to content retrieved in response to a search. For example, a temporary worker hired to find documents for a sales pitch might not have access to documents associated with executives or documents that contain confidential company information. The collaborative search system can manage a workflow that contains steps performed by collaborative participants with varying levels of access to content. For example, a company officer may be the only collaborative participant allowed to search for a particular set of documents as part of a search request, while other collaborative participants may be allowed to search for less restricted documents.

Cloud Gateway

As shown in FIG. **15**, the system can include a “cloud gateway” **1540**, which may include a network attached storage (“NAS”) filer **1505** or NAS head with a limited amount of local storage, and which advertises CIFS/NFS interfaces out to clients **130** and cloud storage sites **115A-N**. The local storage of the NAS filer **1505** of the cloud gateway **1540** provides a way to satisfy incoming data writes from clients **130** quickly, and to buffer or spool data before it is transferred to cloud storage sites **115A-N** or other cloud storage sites **115** (not shown). The cloud gateway **1540** may include functionality to de-duplicate locally stored data before being written up to cloud storage sites **115A-N**, both of which may be done on a fairly rapid or aggressive schedule.

In addition to providing REST-based methods to input and output data from the system, the cloud gateway **1540** may also provide conventional methods of accessing data via a NAS filer **1505** such as via Web-based Distributed Authoring and Versioning (WebDAV) and CIFS/NFS methods, thus making it easy for users and applications to read and write data to cloud storage sites **115A-N** without significant changes to their current mode of working.

Overall, users and applications can specify parameters (e.g., under a storage policy) that dictate to the cloud gateway **1540** the handling of their content—i.e., how long it is retained, should it be encrypted/compressed, should it be deduplicated, should it be indexed and searchable, should it be replicated and if so, how many copies and to where, etc. The cloud gateway **1540** may facilitate the cloud storage system by allowing for metadata to be specified on a per file/object basis or on a data container or bucket basis.

54

Further, the system permits data to be replicated on demand to selected geographies based on access usage patterns, etc. **23. Cloud Gateway Architecture**

FIG. **16** shows a block diagram illustrating a suitable environment for the cloud gateway **1540** that can include a filer or NAS filer **1505** configured to perform data migration to cloud storage sites and other secondary storage. Some details on suitable systems and methods for performing data migration using a NAS filer **1505** will now be presented. Further details may be found in the assignee’s U.S. patent application Ser. No. 12/558,640, filed Sep. 14, 2009, entitled DATA TRANSFER TECHNIQUES WITHIN DATA STORAGE DEVICES, SUCH AS NETWORK ATTACHED STORAGE PERFORMING DATA MIGRATION.

While the examples below discuss a NAS filer **1505**, any architecture or networked data cloud storage site employing the following principles may be used, including a proxy computer coupled to the NAS filer **1505**. The computing system **1600** includes a data storage system **1610**, such as storage operation cell **150**. Client computers **1620**, including computers **1622** and **1624**, are associated with users or servers that generate data to be stored in secondary storage. The client computers **1622** and **1624** communicate with the data storage system **1610** over a network **1630**, such as a private network such as an intranet, a public network such as the Internet, and so on. The networked computing system **1600** includes network-attached storage, such as the cloud gateway **1540**.

The cloud gateway **1540** includes NAS-based storage or memory, such as a cache **1644**, for storing data received from the network, such as data from client computers **1622** and **1624**. (The term “cache” is used generically herein for any type of storage, and thus the cache **1644** can include any type of storage for storing data files within the NAS filer **1505**, such as magnetic disk, optical disk, semiconductor memory, or other known types of storage such as magnetic tape or types of storage hereafter developed.) The cache **1644** may include an index or other data structure in order to track where data is eventually stored (e.g., location in the cloud), or the index may be stored elsewhere, such as on the proxy computer. The index may include information associating the data with information identifying a secondary cloud storage site that stored the data, or other information. For example, as described in detail below, the index may include both an indication of which blocks have been written to secondary storage (and where they are stored in secondary storage), and a lookup table that maps blocks to individual files stored within the cloud gateway **1540**.

The cloud gateway **1540** also includes a data migration component **1642** that performs data migration on data stored in the cache **1644**. While shown in FIG. **16** as being within the NAS filer **1505**, the data migration component **1642** may be on a proxy computer coupled to the NAS filer. In some cases, the data migration component **1642** is a device driver or agent that performs block-level, sub-object-level, or object-level data migration of data stored in the cache, or a combination of two or more types of data migration, depending on the needs of the system. During data migration, the NAS filer **1505** not only transfers data from the cache of the device to one or more cloud storage sites **115A-N** located on the network **1630**, but also to other secondary storage locations **1650**, such as magnetic tapes **1652**, optical disks **1654**, or other secondary storage **1656**. Importantly, the cloud gateway **1540** may also retrieve data from these other secondary storage devices and transfer it to the cloud storage sites **115A-N** (under ILM or other storage policies). The NAS filer **1505** may include various data storage compo-

US 10,248,657 B2

55

nents that are used when identifying and transferring data from the cache 1644 to the secondary cloud storage sites 1650. These components will now be discussed.

Referring to FIG. 17, a block diagram illustrating the components of the NAS filer 1505 component of the cloud gateway 1540, configured to perform data migration, is shown. In addition to the data migration component 1642 and cache or data store 1644, the cloud gateway 1540 may include an input component 1710, a data reception component 1720, a file system 1730, and an operating system 1740. The input component 1710 may receive various inputs, such as via an iSCSI protocol. That is, the cloud gateway may receive commands or control data from a data storage system 1610 over IP channels. For example, the data storage system 1610 may send commands to a cloud gateway's IP address in order to provide instructions to the NAS filer 1505. The data reception component 1720 may receive data to be stored over multiple protocols, such as NFS, CIFS, and so on. For example, a UNIX-based system may send data to be stored on the NAS filer 1505 over an NFS communication channel, while a Windows-based system may send data to be stored on the NAS filer over a CIFS communication channel.

Additionally, the cloud gateway 1540 may include a number of data storage resources, such as a data storage engine 1760 to direct reads from and writes to the data store 1644, and one or more media agents 1770. The media agents 1770 may be similar to the secondary storage computing devices 165 described herein and may similarly be communicatively coupled to one or more SS indices (e.g., SS index 261 and SS light index 204) and deduplication database 297. The media agents 1770 may comprise components similar to those of the secondary storage computing devices 165, such as deduplication module 299, content indexing component 205, network agent 235, media file system agent 240 (including cloud storage submodule 236), as described previously. In some cases, the cloud gateway 1540 may include two or more media agents 1770, such as multiple media agents 1770 externally attached to the cloud gateway. The cloud gateway 1540 may expand its data storage capabilities by adding media agents 1770, as well as other components.

As discussed herein, the cloud gateway 1540 includes a data migration component 1642 capable of transferring some or all of the data stored in the cache 1644. In some examples, the data migration component 1642 requests and/or receives information from a callback layer 1750, or other intermediate component, within the cloud gateway. Briefly, the callback layer 1750 intercepts calls for data between the file system 1730 and the cache 1644 and tracks these calls to provide information to the data migration component 1642 regarding when data is changed, updated, and/or accessed by the file system 1730. Further details regarding the callback layer 1750 and other intermediate components will now be discussed.

In some examples, the cloud gateway 1540 monitors the transfer of data from the file system 1730 to the cache 1644 via the callback layer 1750. The callback layer 1750 not only facilitates the migration of data portions from data storage on the cloud gateway to secondary storage, but also facilitates read back or callback of that data from the secondary storage back to the cloud gateway. While described at times herein as a device driver or agent, the callback layer 1750 may be a layer, or additional file system, that resides on top of the file system 1730. The callback layer 1750 may intercept data requests from the file system 1730, in order to identify, track, and/or monitor data requested by the file system 1730, and may store information associated with these requests in a data structure. Thus, the callback layer

56

stores information identifying when a data portion is accessed by tracking calls from the file system 1730 to the cache 1730.

For example, adding the cloud gateway 1540 described herein to an existing networked computing system can provide the computing system with expanded storage capabilities, but can also provide the computing system with other data storage functionality. In some examples, the cloud gateway 1540 not only provides the storage benefits of a NAS filer 1505, but also includes a data storage engine (e.g., a common technology engine, or CTE, provided by Commvault Systems, Inc. of Oceanport, N.J.), or other functionality. For example, the cloud gateway may perform various data storage functions normally provided by a backup server, such as single instancing, data classification, mirroring, content indexing, data backup, encryption, compression, and so on. Thus, in some examples, the cloud gateway described herein acts as a fully functional and independent device that an administrator can attach to a network to perform virtually any data storage function.

24. Cloud Gateway for Cloud Storage Sites and Deduplication and Policy-Driven Data Migration

As described herein, in some examples, the cloud gateway 1540 leverages block-level, sub-object-level, or object-level data migration in order to provide expanded storage capabilities to a networked computing system. After selecting data for migration, but prior to data migration, the cloud gateway may perform block-level, sub-object-level, and/or object-level deduplication using the methods and/or data structures described previously with respect to FIGS. 1-9. To do so, the cloud gateway 1540 may utilize components or modules within the data storage system 1610 (e.g., a deduplication module 299 and/or a deduplication database 297) and/or utilize components within the cloud gateway itself (e.g., data migration components 1652). In this manner, the cloud gateway may avoid creating unnecessary additional instances of the selected data within secondary storage (e.g., additional instances within cloud storage sites). Additionally, the cloud gateway, may access and apply storage policies as described previously with respect to the system of FIG. 1 to determine to which cloud storage site 115A-N or other cloud storage sites the cloud gateway should migrate the data.

For example, in accordance with a storage policy, the cloud gateway 1540 may utilize more expensive cloud storage sites to store critical documents, and less expensive cloud storage site to store personal emails. As another example, the cloud gateway may implement a storage policy that specifies that a first type of files should be retained for one year in a first target cloud storage site 115A, that a second type of files should be retained for seven years in a second cloud storage site 115B, and that a third type of files should be retained indefinitely in a third cloud storage site 115N. As yet another example, the cloud gateway may implement a storage policy that specifies that a first type of files (e.g., secondary disk copies needed for rapid disaster recovery) be stored only in storage sites 115, including cloud storage sites 115A-N, that can provide sufficient bandwidth, network capacity or other performance to ensure that the time needed to recover a file from the storage device 115 (e.g., cloud storage site 115A-N) is less a specified recovery time objective. As another example, certain data may be migrated or copied only to cloud storage sites 115A-N having sufficient fault tolerance; for example, certain data may be migrated or copied to cloud storage sites that replicate data to various geographic locations to prevent data loss in the event of a natural disaster or similar catastrophic

event. For brevity, the full details of such deduplication and policy-driven storage methods are not repeated here.

The system can perform file system data migration at a file or block level. Block-level migration, or block-based data migration, involves migrating disk blocks from the data store or cache 1644 to secondary media, such as secondary cloud storage sites 1650. This migration process works particularly well with large files spanning many blocks, and is described in detail below. While not shown, file level migration employs similar processes, but is much simpler. Using block-level migration, the cloud gateway 1540 transfers blocks from the cache 1644 that have not been recently accessed from secondary storage, freeing up space on the cache. By tracking migrated blocks, the system can also restore data at the block level, which may avoid cost and time issues commonly associated with restoring data at the file level.

Alternatively or additionally, a cloud gateway 1540 and associated techniques described herein may make secondary disk copies to disaster recovery (DR) locations using auxiliary copy or replication technologies. Additionally or alternatively, a cloud gateway and associated techniques described herein may be used on copies of data created by replication operations such as CDR (Continuous Data Replication) and DDR (Discrete Data Replication).

Referring to FIG. 18, a flow diagram illustrating a routine 1800 for performing block-level data migration in a cloud gateway 1540 is shown. In step 1810, the cloud gateway, via the data migration component 1642, identifies data blocks within a cache that satisfy a certain criteria. The data migration component 1642 may compare some or all of the blocks (or, information associated with the blocks) in the cache 1644 with predetermined criteria. The predetermined criteria may be time-based criteria within a storage policy or data retention policy.

In some examples, the data migration component 1642 identifies blocks set to be “aged off” from the cache. That is, the data migration component 1642 identifies blocks created, changed, or last modified before a certain date and time. For example, the system may review a cache for all data blocks that satisfy a criterion or criteria. The data store may be an electronic mailbox or personal folders (.pst) file for a Microsoft Exchange user, and the criterion may define, for example, all blocks or emails last modified or changed 30 days ago or earlier. The data migration component 1642 compares information associated with the blocks, such as metadata associated with the blocks, to the criteria, and identifies all blocks that satisfy the criteria. For example, the data migration component 1642 identifies all blocks in the .pst file not modified within the past 30 days. The identified blocks may include all the blocks for some emails and/or a portion of the blocks for other emails. That is, for a given email (or data object), a first portion of the blocks that include the email may satisfy the criteria, while a second portion of the blocks that include the same email may not satisfy the criteria. In other words, a file or data object can be divided into parts or portions where only some of the parts or portions change.

To determine which blocks have changed, and when, the cloud gateway 1540 can monitor the activity of the file system 1730 via the callback layer 1750. The cloud gateway may store a data structure, such as a bitmap, table, log, and so on within the cache 1644 or other memory in the NAS filer 1505 or elsewhere, and update the data structure whenever the file system calls the cache 1644 to access, update, or change the data blocks within the cache 1644. The callback layer 1750 traps commands to the cache 1644,

where that command identifies certain blocks on a disk for access or modifications, and writes to the data structure the changed blocks and the time of the change. The data structure may include information such as the identification of the changed blocks and the date and time that the blocks were changed. The data structure, which may be a table, bitmap, or group of pointers, such as a snapshot, may also include other information, such as information that maps file names to blocks, information that maps sub-objects to blocks and/or file names, and so on, and identify when accesses/changes were made.

In step 1820, the cloud gateway 1540 transfers data within the identified blocks from the cache 1644 to a media agent 1770 to be stored in a different data store. The system may perform some or all of the processes described with respect to the system of FIG. 1 when transferring the data to the media agent. For example, before transferring data, the system may review a storage policy as described herein to select a media agent, such as secondary storage computing device 165, based on instructions within the storage policy. In step 1825, the system optionally updates an allocation table, such as a file allocation table (“FAT”) for the file system 1730 associated with the cloud gateway to indicate the data blocks that no longer contain data and are now free to receive and store data from the file system.

In step 1830, via the media agent 1770, the cloud gateway 1540 stores data from the blocks to a different data store. In some cases, the cloud gateway, via the media agent 1770, stores the data from the blocks to a secondary cloud storage site, such as a cloud storage site 115A-N. For example, the cloud gateway may store the data from the blocks in secondary copies of the data store, such as a backup copy, an archive copy, and so on. Although not shown, prior to storing the data from the blocks to a different data store, the cloud gateway, via the media agent 1770, may perform block-level deduplication and/or content indexing, using the methods and data structures described previously with respect to the system of FIG. 1.

Although not shown, prior to storing data from the blocks to a different data store, the cloud gateway 1540 may encrypt and/or compress data as described previously with respect to FIG. 3B. The cloud gateway may create, generate, update, and/or include an allocation table, (such as a table for the data store) that tracks the transferred data and the data that was not transferred. The table may include information identifying the original data blocks for the data, the name of the data object (e.g., file name), the location of any transferred data blocks (including, e.g., offset information), and so on. The location of the transferred data blocks may comprise a URL to a file located on cloud storage site 115A-N. For example, Table 3 provides entry information for an example .pst file:

TABLE 3

Name of Data Object	Location of data
Email1	C:/users/blocks1-100
Email2.1 (body of email)	C:/users/blocks101-120
Email2.2 (attachment)	http://www.cloudstoragesite.com/companyname/remov1/blocks1-250
Email3	http://www.cloudstoragesite.com/companyname/remov2/blocks300-500

In the above example, the data for “Email2” is stored in two locations, the cache (C:/) and an offsite data store

US 10,248,657 B2

59

located on a cloud storage site **115A-N** (<http://www.cloudstoragesite.com/companyname/>). The system maintains the body of the email, recently modified or accessed, at a location within a data store associated with a file system, "C:/users/blocks101-120." The system stores the attachment, not recently modified or accessed, in a separate data store, "<http://www.cloudstoragesite.com/companyname/remov1/blocks1-250>." Of course, the table may include other information, fields, or entries not shown. For example, when the system stores data to tape, the table may include tape identification information, tape offset information, and so on.

Sub-object-based file migration, or sub-object-based data migration, involves splitting a data object into two or more portions of the data object, creating an index that tracks the portions, and storing the data object to secondary storage via the two or more portions. The nature of sub-objects was described previously with respect to the description of deduplication module **299**. As described above, in some examples the cloud gateway **1540** migrates sub-objects of data (sets of blocks) that comprise a data object from the cache **1644** to another storage location, such as to a cloud storage site. In some cases, the data migration component **1642** may include a division component that divides data objects into sub-objects. The division component may perform in a substantially similar fashion to the object division component described previously with respect to the deduplication module **299**. The division component may receive files to be stored in the cache **1644**, divide the files into two or more sub-objects, and store the files as two or more sub-objects in the cache. The division component may update more or more indexes that maintains information to associate particular files with their corresponding sub-objects for that file, the data blocks of the sub-objects, and so on.

The division component may perform different processes when determining how to divide a data object. For example, the division component may include indexing, header, and other identifying information or metadata in a first sub-object, and include the payload in other sub-objects. The division component may identify and/or retrieve file format or schema information from an index, FAT, NFS, or other allocation table in the file system to determine where certain sub-objects of a data object reside (such as the first or last sub-object of a large file). The division component may follow a rules-based process when dividing a data object, where the rules may define a minimum or maximum data size for a sub-object, a time of creation for data within a sub-object, a type of data within a sub-object, and so on.

For example, the division component may divide a user mailbox (such as a .pst file) into a number of sub-objects, based on various rules that assign emails within the mailbox to sub-objects based on the metadata associated with the emails. The division component may place an index of the mailbox in a first sub-object and the emails in other sub-objects. The division component may then divide the other sub-objects based on dates of creation, deletion or reception of the emails, size of the emails, sender of the emails, type of emails, and so on. Thus, as an example, the division component may divide a mailbox as follows:

User1/Sub-object1 Index
 User1/Sub-object2 Sent emails
 User1/Sub-object3 Received emails
 User1/Sub-object4 Deleted emails
 User1/Sub-object5 All Attachments

Of course, other divisions are possible. Sub-objects may not necessarily fall within logical divisions. For example, the

60

division component may divide a data object based on information or instructions not associated with the data object, such as information about data storage resources, information about a target secondary cloud storage site, historical information about previous divisions, and so on.

Referring to FIG. **19**, a flow diagram illustrating a routine **1900** for performing sub-object-level data migration in a cloud gateway **1540** is shown. In step **1910**, the system identifies sub-objects of data blocks within a data store that satisfy one or more criteria. The data store may store large files (>50 MB), such as databases associated with a file system, SQL databases, Microsoft Exchange mailboxes, virtual machine files, and so on. The system may compare some or all of the sub-objects (or, information associated with the sub-objects) of the data store with predetermined and/or dynamic criteria. The predetermined criteria may be time-based criteria within a storage policy or data retention policy. The system may review an index with the division component **815** when comparing the sub-objects with applicable criteria.

In step **1920**, the cloud gateway **1540** transfers data within the identified sub-objects from the data store to a media agent **1770**, to be stored in a different data store. The cloud gateway may perform some or all of the processes described with respect to FIG. **1** when transferring the data to the media agent. For example, the cloud gateway may review a storage policy assigned to the data store and select a media agent based on instructions within the storage policy. In step **1925**, the system optionally updates an allocation table, such as a FAT for a file system associated with the cloud gateway, to indicate the data blocks that no longer contain data and are now free to receive and store data from the file system.

In step **1930**, via one or more media agents **1770**, the cloud gateway **1540** transfers or stores the data from the sub-objects to a different data store. In some cases, the system, via the media agent, stores the data to the cloud storage sites **115A-N**, and/or to secondary storage **1650**, such as magnetic tape **1652** or optical disk **1654**. For example, the system may store the data as secondary copies, such as backup copies, archive copies, and so on. Although not shown, prior to storing the data from the sub-objects to a different data store, the cloud gateway, via the media agent **1770**, may perform sub-object-level or block-level deduplication and/or content indexing, using the methods and data structures described herein.

25. Data Recovery in Cloud Storage Sites via Cloud Gateway Device

A data storage system, using a cloud gateway **1540** leveraging the block-based or sub-object-based data migration processes described herein, is able to restore not only files, but also portions of files, such as individual blocks or sub-objects that comprise portions of the files. Referring to FIG. **20**, a flow diagram illustrating a routine **2000** for block-based or sub-object-based data restoration and modification is shown. While not shown, file level data restoration employs similar processes, but is much simpler. In step **2010**, the system, via a restore or data recovery component, receives a request to modify a file located in a cache of a NAS filer **1505** or in secondary storage in communication with a cloud gateway. For example, a user submits a request to a file system to provide an old copy of a large PowerPoint presentation so the user can modify a picture located on slide **5** of **200** of the presentation.

In step **2020**, the system identifies one or more blocks or one or more sub-objects associated with the request. For example, the callback layer **1750** of the system looks to an index or table similar to Table 3, identifies blocks associated

US 10,248,657 B2

61

with page 5 of the presentation and blocks associated with a table of contents of the presentation, and contacts the cloud gateway 1540 that stored or migrated the blocks on secondary storage.

In step 2030, the system, via the cloud gateway 1540, retrieves the identified blocks or sub-objects from the secondary storage and presents them to the user. For example, the system only retrieves page 5 and the table of contents of the presentation and presents the pages to the user. If some or all of the identified blocks or sub-objects were previously deduplicated prior to being transferred the secondary storage, in order to retrieve the identified blocks or sub-objects, the cloud gateway may utilize the media agent 1770, to “rehydrate” the deduplicated data using the methods described previously with respect to FIG. 12.

In step 2040, the system receives input from a user to modify the retrieved blocks or sub-objects. For example, the user updates the PowerPoint presentation to include a different picture. In step 2050, the system transfers data associated with the modified blocks or sub-objects back to the cloud gateway 1540, where it remains in a cache or is transferred to secondary storage, and updates the table/index. Thus, the system, leveraging block-based or sub-object-based data migration in a cloud gateway, restores only portions of data objects required by a file system.

For example, a user submits a request to the system to retrieve an old email stored in a secondary copy on removable media via a cloud gateway 1540. The system identifies a portion of a .pst file associated with the user that contains a list of old emails in the cache of the cloud gateway, and retrieves the list. That is, the system has knowledge of the sub-object that includes the list (e.g., a division component may always include the list in a first sub-object of a data object), accesses the sub-object, and retrieves the list. The other portions (e.g., all the emails with the .pst file), were transferred from the cloud gateway 1540 secondary storage. The user selects the desired email from the list. The cloud gateway, via an index in the cache that associates sub-objects with data or files (such as an index similar to Table 3), identifies the sub-object that contains the email, and retrieves the sub-object from the associated secondary storage for presentation to the user. Thus, the cloud gateway is able to restore the email without restoring the entire mailbox (.pst file) associated with the user.

As noted above, the callback layer 1750 maintains a data structure that not only tracks where a block or sub-object resides on secondary storage, but also which file was affected based on the migration of that block or sub-object. Portions of large files may be written to secondary storage to free up space in the cache or data store 1644 of the NAS filer 1505. Thus, to the network, the total data storage of the cloud gateway is much greater than that actually available within the cache or data store 1644. For example, while the cache or data store 1644 may have only a 100-gigabyte capacity, its capacity may actually appear as over 20 terabytes, with storage over 100 gigabytes being migrated to cloud-based storage.

26. System Configurations to Provide Data Storage and Management Software as a Service

Alternatively or additionally, the functionality and components of the system described previously may move into the cloud. This solution may be used for software as a service (“SaaS”), for application service providers (ASPs), or for a managed services provider to host and provide data storage and management as an offering, although it can also easily be utilized by a large enterprise to build on top of a private network or cloud. A software as a service (SaaS)

62

model permits a client 130 to utilize a unified and rich set of value-added data management services (e.g. compression, deduplication, content-indexing/search, encryption, etc.) that may be fully independent of which cloud storage providers actually hosting the client’s data. It also provides a mechanism for a client 130 to readily transfer data between various cloud storage sites 115 without being tied to a single cloud storage vendor. A software as a service model also permits clients 130 to utilize data management services and cloud storage on a capacity or utilization basis (e.g., per-gigabyte pricing), without fixed capital expenditures (e.g., expenditures for a set of vendor-specific cloud boxes or a software or hardware license). Under a SaaS arrangement, administrative functions move off-site, since there is no local secondary storage or other hardware at a client’s site and the software (and any software updates) may be pushed to the client 130 as needed and configured on demand. Furthermore, remote monitoring techniques may be employed to further reduce administrative overhead of operating SaaS systems. FIG. 21 illustrates an example of an arrangement 2102 of resources in a computing network that may provide data storage software as a service. As shown, in this arrangement 2102, the storage manager 105 and secondary storage computing devices 165 are in the cloud (e.g., separated from the clients 130 by a network, such as a public WAN, like the Internet). The on-premises components need only include one or more data agents 195 and network client agents 255, which may reside on clients 130. The arrangement 2102 may permit multiple “tenants” to use a single SAAS system 2102 since the various clients 130 may be associated with different entities (e.g., different companies). Data agents 195 utilize network client agents 255 (including HTTP client subagents) to communicate effectively with the storage manager 105 and secondary storage computing devices 165 via their HTTP subagents located within network agents 235.

As described previously, the transport mechanism provided between the HTTP client subagent and HTTP subagents may be cloud-aware and cloud-capable. The HTTP client subagent and HTTP subagents may further be configured to work via firewalls and/or to configure firewalls appropriately. Details regarding managing firewall connections may be found in the assignee’s U.S. patent application Ser. No. 12/643,653, filed Dec. 21, 2009, entitled Managing Connections in a Data Storage System. Alternatively or additionally, data agents 195 may utilize proprietary protocol client subagents configured to facilitate a virtual private network connection running over an HTTPS protocol, or another type of open/secure pipe wrapped in an HTTPS protocol to communicate effectively with storage manager 105 and secondary storage computing devices 165 via their proprietary protocol subagents.

In this arrangement, as described previously, media file system agent 240 may comprise one or more cloud storage submodules 236 that permit the media file system agent 240 to open, read, write, close, and delete data files stored on cloud storage sites and/or otherwise direct cloud storage sites to perform data storage operations.

In this sample arrangement, an on-premises user controlling only the client 130 may benefit from all or some of the system functionalities described previously (e.g., deduplication, content indexing, searching, archiving of data) and yet remain insulated from the details of maintaining and monitoring the data storage architecture on a day to day basis. Those details may move entirely into the domain of the SaaS provider or other network-based or cloud-based service provider, and explained herein.

US 10,248,657 B2

63

27. Object Store

Alternatively or additionally, most or all elements of the system described previously may move into the cloud and be re-configured to allow a cloud storage provider to utilize the system as a data store, such as an object store **2250** shown in FIG. **22**. A large enterprise could also use this system to provide cloud storage and data management to clients within the enterprise and/or outside the enterprise. By exposing REST or other web-based interfaces via a web service layer, users can read, write and manipulate data in an object store **2250**.

In many respects, the object store **2250** provides similar functionality to the systems described previously and may provide additional features. An object store **2250** system may provide value-added services such as retention, deduplication, compression, encryption, content indexing and search, and collaborative searching. An object store **2250** may also provide tiered storage and information life cycle management services. The object store **2250**, like the systems described previously, may also utilize other cloud storage sites as target cloud storage sites **115** that may be used as additional tiers of storage that provide extensible storage capacity.

An operator of the object store **2250** may charge the user of a client **2202** and/or associated entities (e.g., the employer of a user, or another operator or owner of the client **2202**) on a subscription basis, volume basis, a mixed subscription/volume basis, or another pricing structure. For example, an operator may charge a monthly subscription fee to a company for unlimited uploads and downloads to an object store performed by its associated users or clients, so long as the total amount of data stored within the data store at any time during a month does not exceed a certain limit.

As another example, an operator may employ a volume pricing scheme and charge an entity when a user or client that is affiliated with the entity performs various actions using the data store **2250**. The operator may charge an entity a first rate for each unit of data uploaded to the site, and/or a second rate for each unit of data stored in the site for a unit of time (the rate may vary by the type of data cloud storage site used to store the data) and/or a third rate for conducting a content-based search of data stored therein that retrieves information about various objects (e.g., file name, user name, content tags), a fourth rate for conducting a collaborative search operation upon data stored therein, and/or a fifth rate for each unit of data retrieved and/or restored and served back to a client. As a third example, an operator may charge a flat monthly subscription fee to keep a user's account active and additionally charge one or more volume-based rates when the user performs various actions using data store **2250**.

FIG. **22** is a block diagram illustrating components of the object store **2250**. As shown in FIG. **22**, the object store **2250** may comprise a storage manager **105**, one or more object server nodes **2208**, one or more secondary storage computing devices **165**, one or more deduplication databases **297**, and one or more SS indices **261**. An object store **2250** may be communicatively coupled to clients **2202** over a network such as a LAN, MAN, WAN or other network. Clients **2202** may differ from the clients **130** shown in FIG. **1** in that they may not run a dedicated data agent **195** and/or network client agent **255** configured to communicate with the object store **2250**, but instead communicate using existing client-based software components, such as LAN protocols (e.g. Ethernet, SCSI, etc.), WAN protocols (e.g., FTP/HTTP), etc. An object store is communicatively coupled via its secondary

64

storage computing devices **165** to cloud storage sites **115**, including various cloud storage sites **115A-N**, either via LAN, WAN, etc.

As shown in FIG. **22**, each object server node **2208** may comprise an object server agent **2210**, an ingestion database **2212**, and a primary data store **2214**. An object server agent **2210** may be built on Linux for performance and to make it economical to scale the number of object server nodes **2208** as needed. An object server agent **2210** provides a REST interface or other web-based interface to clients **2202** to write, read, retrieve, and manipulate data ingested by the object server node **2208**, and stored therein or in associated secondary cloud storage sites **115**.

Each object server agent **2210** exposes one or more sub-clients of an object server node **2208**. Sub-clients are containers on which default storage policy parameters may be set to dictate the handling or management of data within that container. Individual object-level parameters that a user specifies and provides along with a file/object could optionally override these defaults parameters. Within each sub-client, a number of storage sites can be created, each of which corresponds to a logical point of data ingestion via the REST interface, and may correspond to a particular cloud storage site (e.g., a URL or web directory dedicated to a cloud storage site serving a particular customer or company). Object store **2250** may maintain a system-level (and/or tiered node-level) file system of all data stored within the object store **2250** and/or associated storage devices (cloud storage sites **115**). However, object store **2250** may expose to each particular client (or a particular customer or company) only the subset of the larger file system that corresponds to the client's objects (or a customer's or company's objects). As described herein, object store **2250** may implement these effectively separate file systems in part by utilizing Access Control Lists and/or Access Control Entries.

As an example, a cloud vendor who operates an object store **2250** might assign an entire sub-client to a Web 2.0 customer, who in turn might partition it up into several sites and allocate one to each of its customers. More object server nodes **2208** can be added to the system to scale up the capacity of the object store **2250** and its ability to respond to storage operation requests, while still preserving the ability to address any given site's namespace in the same way. The particular object server node **2208** utilized for the storage of a certain file may be chosen on the basis of the file type and/or other characteristics of the file (e.g. the type of application that created the file). Thus, certain object server nodes may be specific to types of applications (e.g. text-based applications such as word processing applications on one node, image-based applications such as digital image applications on a second node, audio-based applications on a third node, video-based application on fourth node, etc.) As another example, various object server agents **2210** and/or various sub-clients within an object server agent **2210** may each be configured to each handle a different type of object; for example, a first object server agent **2210** may be configured to handle documents, a second object server agent **2210** configured to handle email objects, and a third configured to handle media objects, such as image files and video.

Object server agents **2210** run a web server (such as an Apache or Microsoft IIS web server) and expose a REST interface or other web-based interface to clients **2202**. The object server agents **2210** provide data ingestion or upload points to the object store **2250** for each storage site within each sub-client. Data ingested from a client **2202** by an

US 10,248,657 B2

65

object server agent **2208** may be temporarily stored, cached, or spooled on a primary data store **2214**.

An ingestion database **2212** records information about each data object ingested by its associated object server node **2208**, such as an associated URI or other token that identifies the particular data object, the sub-client and/or site associated with the object, the client **2202** and/or user associated with the object, the time the object was created within the data store, the location(s) of instance(s) of the data object within a primary data store **2214** and/or cloud storage sites **115**, location(s) of deduplication and/or content indexing information pertaining to the object (e.g., deduplication database(s) **297** or SS indices **261** having related information), metadata (including security metadata), default and/or object-level storage policy parameters (such as parameters affecting retention, security, compression, encryption, and content indexing), and an identifier (e.g., a hash). In some examples, the ingestion database may also store content information within the ingestion database **2212** to provide content indexing capability at the object server node. In some examples, the ingestion database **2212** schema comprises tables for sites (e.g. registered sites), security (e.g., document or folder-level security information), objects (or documents), document or object versions, document or object version audit information, deleted document or object versions, storage locations, a document or object cache, and/or archFileReferences. In one example, the ingestion database **2212** is implemented using PostgreSQL, but other examples utilize Oracle, OpenLink Virtuoso, or a similar database management system.

As described previously, data ingested by the object server agent **2210** may be temporarily stored, cached, or spooled on the primary data store **2214**. In one implementation, an ingestion process at the object server node **2008** may run on a prescribed schedule (according to a schedule policy described previously) to process data stored in primary data store **2214**. Using policy parameters, metadata, and/or other information stored in ingestion database **2212**, the object server node **2208** may form logical groups of data objects and request that a secondary cloud storage computing device **165** copy or migrate each logical group of data objects into an archive file or other type of secondary storage format via a secondary storage computing device **165**; each data object in the group is stored in association with related metadata (including Access Control List data and/or other security-related data). Logical groups typically comprise objects having similar retention policies (e.g., similar secondary storage media requirements, similar retention times) and/or similar object types (e.g., all objects in the group are emails; all objects were created using the same application). Logical groups may be formed by applying additional and/or different criteria, such as groups reflecting specific ingestion site(s), user(s) associated with the object, or a company or entity associated with the object. Logical groupings may also be based on policy parameters provided by a client or customer of the object store. Thus, a customer of the object store may provide policy parameters that dictate the logical groupings used. For example a customer might specify that they want a new logical grouping for each back-up cycle performed on their data. As another example, a customer of an object store may specify that they do not want their data commingled with the data of other customers (e.g. the system may consolidate all of that customers data for a particular job or back-up cycle/window to be stored in new containers for that job/cycle/window). In some implementations, an object server node **2208** (or secondary storage computing device **165**) may divide objects into sub-objects

66

(as described previously), form logical groups of data sub-objects, and copy or migrate logical groups of data sub-objects.

As a first example, an object server node **2208** may query an ingestion database **2212** to identify all recently ingested email objects currently stored in primary data store **2214**. Object server node **2209** may then request a secondary storage computing device **165** to process this group of email objects into an archive file stored on a particular cloud storage site **115**. As another example, an object server node **2208** may query ingestion database **2212** to identify all recently ingested objects that are to be stored for 7 years on high-quality tape storage. Object server node **2208** may then request a secondary storage computing device **165** to process this group of objects into an archive file stored on a cloud storage site **115** that provides suitable tape storage.

Unless explicitly proscribed by applicable storage policy parameters, an object server node **2208** may form a logical group that includes data objects from various clients **2202**, each of whom may utilize a different cloud storage site and/or may be affiliated with different entities. In one illustrative example, clients **2202A1**, **2202A2** are affiliated with a Company A and both utilize a first storage site on a first sub-client of a first object server node **2208**. Clients **2202B1** and **2202B2** are affiliated with a Company B and both utilize a second storage site also hosted on the first sub-client of the first object server node **2208**. Assuming the default storage policy parameters of the first sub-client specify that email messages are to be retained on tape for 1 year, then all email objects ingested from all four of these clients may be commingled in a logical group and then stored in a commingled fashion within a single archival tape file scheduled for a one year retention period. The only email objects from these clients that would not be so stored are individual email objects that are associated with different user-specified storage policy parameters (e.g., if a user specified that emails related to or from the finance department should be stored in cloud storage (not tape) and/or stored for a 7 year retention period (not a 1 year period)).

In some implementations, when a secondary storage computing device **165** receives a request to process a logical group of data objects and the metadata associated with these objects, it may handle the request in accordance with the process of FIG. 3B. That is, the secondary storage computing device **165** may content index each object in the group, perform object-level, sub-object level and/or block-level deduplication on the group, and/or encrypt the data and metadata. As a result of the processing, the secondary storage computing device **165** will also store each of the various objects in logical association with its related metadata (including ACL or other security metadata). During this process, described previously, the secondary storage computing device **165** may build indexing information within a content index or another index (e.g., SS index **261**) and/or deduplication information (e.g., within deduplication database **297**). By storing objects with similar retention policies in logically grouped archival files, the system may efficiently prune or eliminate data from the object store **2250** and/or more efficiently perform ILM within the Object store **2250**, since the various objects within each archival file may have similar dates for deletion or migration.

During the deduplication processing of a logical group, the secondary storage computing device **165** may perform lookups on one, some, or all of the deduplication databases **297** within the object store **2250**. In one example, during deduplication, a secondary storage computing device **165** only performs lookups on one deduplication database **297**,

US 10,248,657 B2

67

which may decrease the time required for deduplication (and/or pruning and/or data restoration) but increase the volume of data stored within the data store. In another example, during deduplication, a secondary storage computing device 165 performs lookups on all deduplication databases 297 within an object store 2250, which may increase the time required for deduplication (and/or pruning and/or object restoration) but decrease the volume of data stored within the data store.

Note that deduplication of data objects in a logical group may occur across clients 2202 and/or across various companies. Returning the prior example, if client 2202A1 and client 2202B2 (from two different companies) both receive a particular email message and associated large attachment, secondary cloud storage site 165 may store only one instance of the email data object and attachment (although it stores and associates the instance with two different sets of metadata, one set for client 2202A1 and one set for client 2202B2). Thus, by storing data received from multiple clients, even associated with different and independent companies, the system can realize greater deduplication over what either client would realize individually. Once cross-client or cross-company deduplication occurs, if a particular client or company requests the deletion of a shared object (or shared sub-object or block), the system will not necessarily delete the physical copy of the shared object (or sub-object or block). Instead, the system may simply update one or more indices or databases such as a deduplication database (e.g., by removing a link, URL or other pointer to a physical copy), delete the file name from a file allocation table (FAT) or similar file system data structure, etc. In this way the client or customer who “deleted” the object no longer has access to the object and no longer sees the object as part of the file system that is exposed to them by the object store.

Additionally in this example, under the deduplication processes described previously, even if the two identical email objects were ingested by an object server node 2208 at different times (e.g., a month apart), when a second copy eventually reaches a secondary storage computing device 165, it still might not result in a second instance being created. This result occurs because during the deduplication process, a deduplication module 299 on a secondary storage computing device 165 might detect an instance of the object in a deduplication database 297. However, the system may alternatively determine that the first version, while identical, is too old and could have been stored on storage medium that may be degrading, and thus the system may store the second version it receives years later.

As described previously, when a media file system agent 240 performs the process shown in FIG. 3B it will typically result in the storage of one or more aggregated or containerized archive files. The individual data objects of a logical group are not stored as individual files on a file system of a cloud storage site 115. As described previously, by containerizing data, the object store 2250 may thus reduce the limitations posed by file system scalability by reducing the strain on the namespace of the object store 2250. The generation of these archive files also generates catalogs (e.g., deduplication databases 297, SS indices 261, and/or other information) that makes it easier to access, search for, retrieve, or restore a single object even from the aggregated archive form. Further details on archive files may be found in the assignee’s U.S. Patent Publication No. 2008-0229037, filed Dec. 4, 2007, entitled SYSTEMS AND METHODS FOR CREATING COPIES OF DATA, SUCH AS ARCHIVE COPIES.

68

When a client 2202 or application running on a client 2202 checks in or stores an object into an object store 2250, an object server node 2208 may serve it a unique Universal Resource Identifier (“URI”) or token that points to or identifies the object, which the client 2202 or application may store locally on the client side. This token or URI may be globally unique to all objects within the object store 2250. Alternatively, it may be unique with respect to all objects stored by a single client 2202, ingested by a particular object server node 2208, sub-client and/or site, and/or unique with respect to another factor. In this way, the URI in conjunction with other information (e.g., a user’s login information) may still uniquely identify a particular data object.

To provide verification to a user of the integrity of files stored in an object store 2250, an object store can optionally generate a unique identifier such as a hash (or probabilistically unique identifier) using a particular identifier-generation algorithm for each data object ingested and return that identifier to a calling application on a client 2202 at the time of ingestion. When an application on the client 2202 later retrieves the same data object, a client-side application can use the same identifier-generated algorithm to compute a hash for the retrieved object. If this newly computed identifier matched the identifier returned during ingestion, it would assure the client that the data object had not been modified since it was originally ingested. In addition, an object store 2250 may run similar periodic data verification processes within the object store 2250 asynchronously to ensure the integrity of the data stored therein. Further details may be found in the assignee’s U.S. Patent Publication No. 2009-0319534, filed Jun. 24, 2008, entitled APPLICATION-AWARE AND REMOTE SINGLE INSTANCE DATA MANAGEMENT (U.S. patent application Ser. No. 12/145,347).

Optionally, data objects may be ingested inline into multiple archive files on separate object server nodes 2208 (for redundancy or other reasons). Also, in one example, geographically separate replication may be configured per cloud storage site, which allows the system to serve up objects from a remote location (which may include continuous data replication technology), for fault tolerance (because separate power grids, long-haul communication links, etc. would be used), etc.

An object store 2250 may also optionally make a copy of data on removable media such as tape to enable secure offline storage. Alternatively or additionally, an object store may make secondary disk copies to disaster recovery (DR) locations using auxiliary copy or replication technologies as noted herein.

Each site within an object store 2250 may be protected via security policies that limit which users or clients 2202 have access to the site (and/or to particular objects stored within the site). As described previously, a system may include mechanisms to permit authentication (e.g., by the use of registered username and password combinations and/or similar known authentication methods). A system may also enable customers to specify and store access privileges, including privileges for object access within the object store 2250. As described previously, user-level security and other metadata may be provided and stored along with the object.

For example, an object may be stored with a provided ACL containing Access Control Entries (“ACE”). An ACL contains a list of users and/or groups that are allowed to access a data object, type of data object, or resource containing a data object. Each ACE may specify a user, group, or other entity that has access to the data object associated with the ACL. In some embodiments, an ACL may contain

US 10,248,657 B2

69

a list of users or groups that are specifically denied access to a data object. To implement user-level security, when a user, system, or process attempts to access a data object on an object store **2250** (or related information or metadata, such as a file name), the object store **2250** may access and parse an ACL and any associated ACEs or other security data related to the data object to determine whether the user has the appropriate access level to access the object or its related information. Further details on such security and access control may be found in the assignee's U.S. Patent Publication No. 2008-0243855, filed Mar. 28, 2008, entitled SYSTEM AND METHOD FOR STORAGE OPERATION ACCESS SECURITY (U.S. patent application Ser. No. 12/058,511).

When an application running on a client **2202** requests the retrieval of a data object stored in the object store **2250**, the client may present a URI (or other token) back to the object server node **2208**. Before the object server **2250** returns the data object (and/or provides other related information or metadata to the user, such as the file name of the data object), the object server (e.g., via the object server agent **2210**) may parse the ACL or other security information to confirm that returning the object (or providing other information) is in conformance with the object's security settings and/or previously defined policies stored in the storage manager. If the user of the client **2202** is properly authenticated, and the user has sufficient access rights to the object (as determined by the ACL or other security information stored in conjunction with the object), the user will be able to retrieve the data object. In this manner, the object store **2250** ensures sufficient privacy between various clients **2202A1**, despite the fact that their objects may be commingled in the primary data store **2214** and cloud storage sites **115**.

A web-based portal may be provided by the object store to readily allow a user to authenticate interactively and browse, view, and restore their data as well. For example, a web-based portal may permit a user to log on to the system, and may then present a user with an interface that presents to them various data objects associated with the user. For example, it may present objects that were ingested from the user's client **2202**, and/or objects ingested from some clients from the user's entity, and/or objects associated with a collaborative search in which the user is a participant. The interactive interface will also support search capabilities, end-user tagging of data, and the ability to classify data into folders ("review sets") for future reference.

Data indexing capabilities, described above, may be incorporated into an object store **2250** to permit policy-based searches of content or other information relating to data objects, that have been indexed. Such data indexing and classification permits the object store **2250** to offer "active management" of the data to an administrator of the system. For example, an administrator can define the actions to be performed on data based on criteria pertaining to the data—e.g., tag, check into an ECM system, restore into a review set for a knowledge worker to review later, etc. In one example, indexing capabilities may also permit users to conduct collaborative searching and collaborative document management of objects within the object store **2250** as described previously.

28. Object Store Methods

In one implementation, an object store **2250** may avoid the system costs associated with uploading and storing an unnecessarily duplicative copy of an object during a data storage request by a client **2202**. FIG. 23 shows a first process **2300** for managing a request to store an object within an object store **2250**, including apportioning the

70

storage cost of the object. The process **2300** may result when a calling application on a client **2202** requests that an object server agent **2210** store a particular object.

The process **2300** begins in block **2305** where an object server node **2208** receives an identifier (e.g., a token, URI or hash) for an object and metadata associated with the object (including, e.g., object-level security, content tags, and/or storage policy parameters). For example, a calling application on the client **2202** may generate a hash identifier for an object and send that identifier to object store **2250** along with metadata. At optional block **2310** the object server node **2208** performs a lookup of the received identifier in one or more deduplication database(s) **297** to ascertain whether the object has already been ingested and processed by object store **2250** (or ingested or processed by particular object server node(s) **2208**, particular storage site(s), or particular secondary storage computing device(s) **165**, such as those secondary storage computing device(s) **165** associated with a particular object server node **2208**). Alternatively or additionally, an object server node **2208** performs a lookup of the received identifier in one or more ingestion databases **2212** within data store **2250** to ascertain whether the object has already been ingested by object store **2250** (or ingested by particular object server node(s) **2208**, particular sub-client(s), or particular storage site(s)). Alternatively or additionally, the object server node sends the received identifier to one or more cloud storage sites to see if a copy of the object has already been stored therein.

At optional decision block **2315**, the system uses the information acquired at block **2310** to determine if the system currently has the object stored in a manner that is consistent with the storage policy parameters applicable to the object. If it does, the process proceeds to block **2355**, otherwise it proceeds to block **2320**. For example, if the system has only one copy of the object stored in tape storage, but the calling application on the client **2202** has specified that the object should be stored on disk storage, the process may proceed to block **2320**.

If object store **2250** already has the object stored in an appropriate manner, at block **2355**, the object server node **2208** updates deduplication database **297** to reflect how the new request refers to previously stored blocks. For example, the system may increase reference counts in a primary block table and may add additional entries to a secondary block table within deduplication database **297** to reflect how the new request refers to previously stored blocks. In some implementations, the system may additionally or alternatively update an object-level deduplication database **297** (e.g., by incrementing an object-level reference count in an object-level index within the deduplication database).

At block **2355**, the object store **2250** may not request a new copy of the object, saving the time and system cost associated with uploading the object anew, and may instead simply update a deduplication database **297**. For example, if a cloud storage site already has a copy of an object stored therein, at step **2355**, the object store may add a link or URL to a previously stored copy in the deduplication database **297** and/or elsewhere. The process then proceeds to block **2325**.

If optional blocks **2305-2315** are not performed by the system, the process begins instead at block **2320**.

At block **2320**, object server node **2208** requests the object from client **2202**. If object server node **2208** has not already received metadata, it also requests metadata from client **2202** at block **2320**. The process then proceeds to block **2325**. Alternatively, if at decision block **2315**, the object server node **2208** determines that the object store

US 10,248,657 B2

71

2250 currently has the object in storage, but it is stored in a manner that is inconsistent with applicable storage policy parameters, object server node **2208** may instead retrieve or request a copy of the object from another system component (e.g., a primary data store **2214** or a cloud storage site **115**) and if necessary, request metadata from client **2202**.

At block **2325**, after receiving the object and/or its metadata, the system stores these in the primary data store **2214**. If object store **2250** already has a copy of the object stored in an appropriate manner, at block **2325** the system may store one or more pointers, links, or references to the object and/or its constituent blocks (e.g., a pointer to a dehydrated instance of the object within object store **2250** or cloud storage site **115**, or a pointer or reference to an entry in a deduplication database **297**) in the primary data store **2214** instead of storing a copy of the object. At block **2325**, an object server node **2208** may also generate a URI for the object, update an ingestion database **2212** to reflect information about data object (as described previously), and may return a URI or other token to client **2202**. Additionally or alternatively, an object server node **2208** may also generate and return an identifier (e.g., a hash) for the object to provide later validation to the client **2202**. Object server node **2208** may also store an identifier for the object in ingestion database **2212** and/or deduplication database **297**.

At block **2330**, during a scheduled ingestion process described previously, object server node **2208** may associate the object (and its metadata) with a logical group of objects (logical groupings are described further herein). Object server node **2208** may further request a secondary storage computing device **165** to process the logical group by copying or migrating each logical group of data objects into a compressed, deduplicated or “dehydrated” archive file that may employ data structures such as those shown in FIGS. **5** and **8**.

At block **2335**, a secondary storage computing device **165** performs content indexing of the object in the manner described previously with respect to FIG. **10**. At block **2340**, a secondary storage computing device **165** performs deduplication of the object using one or more of the deduplication methods and data structures described previously. In one example, deduplication may be file or block-level deduplication. In other examples, the deduplication may be object-level or sub-object level deduplication. During deduplication at block **2340**, the system may perform lookups on or otherwise examine one, several, or all deduplication databases **297** within object store **2250** to determine the number of instances of the object that are currently stored and/or the number of instances of each block in the object that are current stored. Thus, the scope of deduplication within an object store **2250** may be quite limited or quite broad. In one example, a deduplication process only utilizes deduplication databases **297** associated with the same object server node **2208** that received or ingested the object. A deduplication database **297** is associated with an object server node **2208** if the deduplication database has any entries reflecting a storage operation initiated by the same object server node **2208**.

At block **2345** the system stores a dehydrated form of the object within an archive file, which may also comprise data relating to any or all of the objects in the logical group. As illustrated previously, the precise dehydrated form of an object within an archive file will depend on the type of deduplication performed and whether some or all of the object’s content had previously been stored. For example, if block-level deduplication is performed upon the object and a prior instance of the object was already appropriately

72

archived, the dehydrated form of the object may be represented within the archive file by metadata and one or more pointers or similar references. For example, during deduplication, if a cloud storage site already has a copy of an object stored therein, at step **2345**, the object store may store in a container file, a link, URL or other pointer to a previously stored copy. If instead, block-level deduplication is performed upon the object but a prior instance of the object was not already appropriately archived, the dehydrated form of the object within the archive file may comprise metadata, pointers/references to some blocks stored previously, and new copies of some other blocks within the object.

At optional block **2360**, the system may apportion the cost of storing the object between one or more clients or their related entities. Stated conversely, at block **2360**, the system may attempt to apportion any cost savings resulting from the avoidance of unnecessary storage within the data store and/or unnecessary uploads to the object store **2250**. For example, if two different clients **2202** from two different companies both request that an object store **2250** provide storage of the same data object, the two companies may receive adjusted pricing for their requests to reflect the cost savings realized by the system during deduplication. As described previously with respect to FIG. **22**, in the event that some or all of the blocks of the data object were previously stored appropriately within the storage operation cell **2250**, the deduplication at block **2340** may reduce the amount of data needed to process a new request to store the same data object. Thus, block **2340** may reduce the amount of data storage needed to accommodate a storage request. Additionally, if the system performs the optional identifier lookup shown in blocks **2305-2315** and the process proceeds to block **2355**, the system avoids the cost of receiving the data object (e.g., ingestion bandwidth of an object server agent **2210** used and/or the system resources needed to transfer the object into and out of a primary data store **2214**).

To apportion cost savings, the system may utilize or mine the data stored in deduplication databases **297**, SS index **261**, management index **211**, and/or ingestion databases **2212**. As described previously, these databases correlate client **2202** information with data ingested into and stored by the object store **2250**, such as the time of creation, deduplication information, deletion dates, and storage locations. Thus, the system may use these databases to determine which storage requests initiated by a particular client **2202** were processed via direct ingestion of an object from the client **2202**, in contrast to those storage requests initiated by the client that were able to utilize previously stored instances of an object or some of its blocks. Such a determination permits the system to determine where cost savings have occurred. When apportioning costs, the system may utilize a sliding ratio that is selected using criteria such as the size of a shared data object, the quantity and/or quality of total data stored on the object store by a particular company or client, the terms of a service contract or agreement between a particular company and an operator of an object store, the storage policy for the company, and/or any other suitable criteria.

In one example, a first client **2202A** associated with a first company uploads a new object to an object store **2250**, and later a second client **2202C** associated with a second company sends an identifier (hash) of the same object to the object store and requests storage of that object. In this example, a second upload of the object itself may be avoided (i.e., the process of FIG. **23** proceeds to block **2355**) and a second copy of the object within the object store **2250** may be avoided. In this example, the system may initially charge

US 10,248,657 B2

73

the first company a first non discounted rate for the upload of the object (e.g., a rate based on its size) and a second non discounted rate for the storage of that object (e.g., a rate based on the object's size and the duration and quality of storage used to store it). At a later time, the system may charge the second company a third discounted rate for their requested upload of the object (e.g., a rate based on its size) and a fourth discounted rate for the storage of that object (e.g., a rate based on the object's size and the duration and quality of storage used to store it).

Additionally or alternatively, the first company may receive a credit or rebate to its account to reflect some or all of the cost savings realized from avoiding a second upload; this credit or rebate may be for an amount that is different from (e.g. less than) the second client's third rate. Additionally, after the second client requests storage, so long as both the first and second clients have effective access to the data object (e.g., their "virtual copy" of the object has not been eliminated due to a retention policy and the client has not requested its deletion), one or both companies may receive a discounted or reduced storage rate. For example, the first company may receive a storage rate lower than the second non-discounted rate that was originally charged.

In a second example, a first client **2202A** associated with a first company uploads a first object that is new to the object store **2250**, and later a second client **2202C** associated with a second company sends an identifier (e.g., a hash) of a similar second object and requests storage of the object. A second object is similar to a first object if it shares one or more blocks in common with the first object. In this example, a second upload of the object itself is not avoided (e.g., the process proceeds to block **2320**), since the two objects have different identifiers. However, block-level deduplication (e.g., at block **2340**) may reduce the amount of new data needed to store the second object. After the second client requests storage, so long as both clients have effective access to the common blocks (e.g., their "virtual copy" of the blocks has not been eliminated due to retention policies and the client has not requested deletion), one or both of the two companies may receive a reduced storage rate for the common blocks.

In a third example, cost apportionment is not tied to a particular storage request, but rather occurs in an aggregated way. For example, the system may periodically (e.g., monthly) determine what percentage of blocks uploaded directly from a first company's clients **2202** are referenced by another company's deduplication database entries. The system might then provide a rebate to the first company's account, offer lower rates to the first company for another future period (e.g., the next month), apportion costs that month between the two companies so that each company's bill is less than what it would have been if each had stored its own copy, etc.

In a second implementation, an object store **2250** may avoid the system costs associated with uploading and storing unnecessary duplicate copies of data blocks when processing a data storage request by a client **2202**. FIG. **24** shows a second process **2400** for managing a request to store an object within an object store **2250**, including apportioning the storage cost of the object. The process **2400** of FIG. **24** is similar to process **2300** of FIG. **23**, however, in process **2400**, the system may avoid the costs associated with uploading redundant blocks, not just redundant objects, by performing block-level deduplication at substantially the same time as data ingestion. In this implementation, during process **2400** the system may cache or store a logical group of objects in an archive file stored in the primary data store

74

2214 that reflects a dehydrated form of the objects (i.e., an archive file that utilizes data structures similar to those shown in FIGS. **5** and **8**). Later, during a scheduled ingestion process, the archive file may be transferred or copied to one or more secondary cloud storage sites **115**.

Alternatively, during process **2400**, the object store **2250** may write a dehydrated form of data objects directly to an archive file located in a secondary data store **115** by utilizing secondary storage computing device **165**. As described previously, an archive file may comprise one or more volume folders **802** that further comprise one or more chunk files **804**, **805**. The chunk folders may further comprise one or more of each of the following: metadata files **806**, metadata index files **808**, container files **810**, **811**, and container index files **812**.

The process **2400** begins at block **2405**, where the system receives object metadata, identifies a logical group, and identifies an archive file for storing a dehydrated form of the object. At block **2405**, the system may identify a logical group for the object by using the received metadata (e.g., reflecting the type of object, the storage policy parameters, and/or security information), and/or other information (e.g., the identity of the client **2202** making the storage request) to identify a logical group of objects having similar storage policy parameters, similar object types, and/or other similarities. Once a logical group is identified, the system identifies an archive file utilized by the system to store the logical group in a dehydrated form. The archive file may be located in primary data store **2214** or on a secondary cloud storage site **115**. If a suitable archive file does not already exist in primary data store **2214** (e.g., because archive files were recently migrated from primary data store **2214** to secondary cloud storage sites **115**), the system may create a new archive file in primary data store **2214** for the logical group. Alternatively, the system may create a new archive file in a secondary cloud storage site **115** for the logical group.

At optional blocks **2407-2415**, the system receives an object identifier and performs a lookup of the object in deduplication database(s) **297** to determine whether the object store **2250** already has a copy of the object appropriately stored within the object store. Blocks **2407-2415** are performed in the same manner as blocks **2305-2315** described previously with respect to FIG. **23**. If optional blocks **2407-2415** are not performed, the process **2400** proceeds directly to block **2435**.

If at decision block **2415** the system determines that object store **2250** does have a copy of the object appropriately stored therein, then at block **2420** the system updates one or more deduplication databases **297** to reflect how the identified archive file refers to previously stored blocks. For example, the system may increase reference counts in a primary block table. As another example, the system may add additional entries to a secondary block table within deduplication database **297**. For example, if a cloud storage site already has a copy of an object stored therein, at step **2415**, the object store may add in a deduplication database **297** and/or elsewhere, links or URLs to previously stored blocks. At block **2425**, the system may content index the object. To do so, the system may associate the new storage request with content indexing information previously derived and/or associate the new storage request with metadata provided. Alternatively or additionally, the system may restore all or part of the data object using the processes described previously and content index a restored data object and/or a restored portion of the data object. The system may store some or all of the content index informa-

US 10,248,657 B2

75

tion in the SS index **261** and/or ingestion database **2212**. The process then proceeds to block **2430**.

At block **2430**, the system updates the identified archive file to reflect the storage request. To do so, the system may (1) add the received metadata to a metadata file (2) add links, references, or pointers within the metadata file that point or refer to previously stored blocks, and (3) update a metadata index file. If all of the blocks in the object were previously stored in an appropriate manner, the system may not need to add any additional blocks to a container file. For example, if a cloud storage site already has a copy of an object stored therein, at step **2345**, the object store may store in a metadata file, metadata index file, or another container file, links or URLs to previously stored blocks.

If optional blocks **2407-2415** are not performed or if, at decision block **2415**, the object store does not have a copy of the object appropriately stored therein, the process proceeds to the loop shown at block **2450**, where the system performs blocks **2440-2470** for each block within the object. At block **2440**, the system receives a block identifier. At decision block **2445** the system determines if the system already has an appropriately stored copy of the block by querying one or more deduplication databases **297**. During block **2445**, the system may perform lookups on or otherwise examine one, several, or all deduplication databases **297** within object store **2250** to determine the number of instances of the block that are appropriately stored. Alternatively or additionally, the system sends the received block identifier to one or more cloud storage sites to see if a copy of the block has already been stored therein. Thus, the scope of block-level deduplication within an object store **2250** may be limited or broadened.

If the system does have a copy of the block appropriately stored, then the system at block **2450** updates deduplication databases **297** to associate the current storage request with that block. For example, the system may increment a reference count in a primary block table and add an additional entry to a secondary block table. The process then continues to block **2455**, where the system updates the identified archive file by (1) adding received metadata to a metadata file and/or (2) adding a link, reference, or pointer within the metadata file that points or refers to a previously stored copy of the block. For example, if a cloud storage site already has a copy of a block stored therein, at step **2325**, the object store may add in a metadata file or another container file, a link or URL to a previously stored copy. The process then proceeds to decision block **2470**.

If the system does not have a copy of the block appropriately stored therein, then the system proceeds to block **2460**, where the system requests a copy of the block from the client **2202**. Once the block is received, at block **2465**, the system stores the block in a container file within the identified archive file and otherwise updates the archive file. For example, the system may update a metadata file **806** with a link to the newly stored block and with received metadata. The system may further update deduplication databases **297** by adding a new entry to a primary block table and/or adding an additional entry to a secondary block table.

As shown at decision block **2470**, the sub-process of blocks **2440-2465** repeats so long as there are additional blocks within the object that require processing by the system.

The process **2400** then proceeds to block **2475**, where the system content indexes the object. During content indexing, the system may simply index the object using received metadata (e.g., using content tags provided as metadata by a user). Alternatively or additionally, the system may restore

76

all or part of the data object using the processes described previously and content index a restored data object and/or a restored portion of the data object. The system may store some or all of the index information in the SS index **261** and/or ingestion database **2212** before proceeding to block **2480**.

At block **2480**, the system updates ingestion database **2212** to reflect the processed storage request and received metadata, and returns a URI to the requesting client **2202**.

At optional block **2485**, the system may apportion costs among clients or their related entities in a manner similar to that described previously with respect to FIG. **23**. When apportioning costs, the system may utilize a sliding ratio that is selected using criteria such as the size of a shared data object/block, the quantity and/or quality of total data stored on the object store by a particular company or client, the terms of a service contract or agreement between a particular company and an operator of an object store, storage policy requirements, and/or any other suitable criteria. In one example, a first client **2202A** associated with a first company uploads a first object that is new to the object store **2250**, and later a second client **2202C** associated with a second company sends an identifier (e.g., a hash) of a similar second object and requests storage of the object. The second object is similar to a first object because it shares a set of blocks in common with the first object. In this example, via the process **2400** shown in FIG. **24**, a second upload of the common blocks is avoided. Furthermore, block-level deduplication (e.g., at blocks **2440-2465**) may reduce the amount of new data needed to store the second object. In this example, the system may initially charge the first company a non discounted first rate for both the upload of the object (e.g., based on its size) and a non discounted second rate for the storage of that object (e.g., based on the object's size and the duration and quality of storage used to store it). At a later time, the system may charge the second company a reduced third rate for its request to upload the object to reflect cost savings realized by avoiding a second upload of common blocks. Additionally or alternatively, the first company may receive a credit or rebate to its account to reflect some or all of the cost savings realized from avoiding a second upload; this credit or rebate may be for an amount that is different from the second client's third rate or discount. After the second client requests storage of the second object, so long as both clients have effective access to the common blocks (e.g., their "virtual copy" of the common blocks has not been eliminated due to retention policies and the client has not requested deletion of an associated object), one or both of the two companies may receive a reduced storage rate for the common blocks.

Process for Cost-Balancing Cloud Storage

FIG. **27** is a flow diagram illustrating a process **2700** for identifying suitable storage locations for a set of data objects subject to a storage policy. Process **2700** may be performed by the systems of FIGS. **1**, **2**, **15**, **16**, **21**, and **22** and/or other suitable systems. The process **2700** begins at block **2705** when the system accesses the storage policy applicable to the set of data objects. This storage policy may define different classes of storage devices **115**. For example, the storage policy might define "first-class storage" as any local storage device having magnetic disk or otherwise faster-access storage media and a first cloud storage site that satisfies certain criteria (e.g., has high bandwidth for faster uploads and/or downloads and/or utilizes RAID or similar methods that improve the fault-tolerance of the site), and "second-class storage" as a second cloud storage site that may have greater latencies or lower fault-tolerance and any

US 10,248,657 B2

77

local storage device having magnetic tape or otherwise slower data storage. Additionally, the storage policy may also define different categories of data objects (e.g. functional categories such as email objects, audio objects, video objects, database objects, document objects, etc.) and may require different classes of storage for each.

At block **2710**, the system logically groups the various data objects and determines the storage requirements of each group. Typically the system groups the set of data objects so that each group requires a particular class of storage. However, the system may group the various data objects by any other logical grouping such as groups based around functional categories, or to improve the possibility of realizing deduplication benefits. The particular grouping used by the system will be chosen to conform to the storage policy. Logical groupings are described in greater detail herein.

The system may first utilize the storage policy and the management light index **245**, the management index **211**, the SS index **261**, the SS light index **247**, deduplication database **297** and/or metabase **270** to determine the number of bytes, kilobytes, gigabytes, terabytes or similar units required to store each individual data object, and any other requirements necessary to conform to the storage policy. For example, the system might determine that a particular data object requires 25 megabytes of first-class storage. The system may next determine the aggregate storage requirements for each group of data objects. For example, the system may determine that a first group of data objects requires an aggregate 200 gigabytes of first-class storage and a second group of data objects requires an aggregate 450 gigabytes of second-class storage. The aggregate storage requirements determined by the system may reflect the effect of deduplication; for example, the system may utilize deduplication database **297** to determine the size of an archive file created in part by block-level deduplication.

The system then performs blocks **2712-2740** for each group of data objects to determine the appropriate storage location of the various data objects in the group. At block **2712**, the system identifies the storage devices **115** (including cloud storage sites **115A-N**) that may be suitably employed to store the group of data objects. To determine the list of potential storage devices **115** (referred to as "candidates"), the system may access storage device class definitions in the storage policy. The system may also access data regarding storage devices **115** stored in the management index **211**, secondary storage computing devices **265** and/or storage devices **115**. For example, if the group of data objects requires first-class storage, the system may query the management index **211** to determine which local magnetic storage devices **115** have sufficient storage capacity to accommodate the group of data objects.

At block **2715**, the system may transmit a request for quotes to candidate cloud storage sites (which may be operated by independent organizations) identified at block **2712** (or other appropriate types of data storage service providers accessible via the network). To do so, the system may initiate communications via the network agent **235**. For example, the system will request a quote from each cloud storage site by initiating an HTTP connection with the cloud storage site and sending the request via one or more HTTP messages. This request for quotes may include information such as: the amount of storage space required, a unique identifier associated with the request, an identifier associated with a prior request made or a quote received from the site (e.g., in the case of a counter offer), information that identifies the system making the request (or identifies a related entity, such as a billing party), how the data will be

78

accessed once stored or how often (i.e., accessibility of data, including desired data transfer rates), a suggested or required upload time window or deadline, estimated storage lifetime of the objects, suggested pricing rate(s), the type of storage medium desired (e.g., tape or optical or magnetic media), maximum pricing rate(s), suggested download, upload, and/or storage pricing rates (and/or a promotional code or similar indicator of a pricing rate package), and/or any other information suitable for requesting a storage quote.

Alternatively, or additionally, the system may obtain estimated storage costs for one or more cloud storage sites by sending similar requests for quotes to one or more third-party sites that provide binding, non-binding and/or informational storage quotes (e.g., a website operated by a data storage dealer-broker or a site that aggregates information regarding cloud storage costs). The format and content of the request may be customized to each site and may be dictated by an API set utilized by a particular cloud storage or third-party site. Alternatively or additionally, the system may estimate the storage costs for a candidate cloud storage site by accessing historical, projected or other cost information stored within the storage manager **105** or elsewhere in the storage operation cell **150**.

At block **2720**, the system may receive one or more quotes from one or more cloud storage and/or third-party sites. For each cloud storage site, the system may receive no quote, a single quote, or several quotes covering various storage options. Each quote may include information such as: one or more pricing rates, the accessibility of stored data, identifiers or tokens associated with the quote, time windows during which data may be transmitted or retrieved, an acceptance window during which the quote would be honored by the site, etc. The quote may provide various pricing rates for different types of data operations. For example, the quote may specify a first rate for an initial upload to the site, a second rate for downloads from the site, and a third rate for searching or accessing the data, a fourth rate for continued storage and maintenance of the data on the site (e.g., a rate charged for each gigabyte stored per month), maximum storage space allotted, maximum or minimum storage lifetime; and so forth. The format and content of the quote may be different for each cloud storage or third-party site and may be dictated by an API set (or similar) utilized by a particular cloud storage or third-party site. The system may perform additional blocks, such as data extraction, to create a uniform set of data for all of the received quotes.

At optional block **2725**, the system may access other historical or projected data pertaining to storage device candidates, including optical, tape or magnetic disk storage device candidates located locally within the storage operation cell **150**. In some embodiments, the system may access historical or projected operating costs of each candidate that may be stored in management index **211**, secondary storage computing devices **265**, or elsewhere in the storage operation cell **150**. In still other embodiments, the system may access data relating to: current or projected power consumption, current or projected power rates, acquisition cost of the storage devices, mean operating time, mean repair time, mean data access rates, or similar performance and cost metrics that may be stored in the management index **211**, secondary storage computing devices **265** or elsewhere.

At block **2730**, the system may evaluate the cost of storing the group of data objects on some or all of the storage device candidates (the "storage cost"). The storage cost associated with a particular storage device may refer simply to the estimated monetary expense associated with uploading the

US 10,248,657 B2

79

group of data objects to the storage device and/or maintaining it there for its estimated lifetime (or other time period).

Alternatively or additionally, the “storage cost” of a certain storage device candidate may refer more generally to the value of a numerical cost function that may take into account several variables. Non-exclusive examples of cost function variables include: historical or projected information pertaining to storage device candidates; any quoted pricing rates; the amount of storage required; the network load associated with uploading and/or downloading the data to a site; projected data access costs; other accessibility metrics; site reliability, quality or reputation; geographical location of a candidate; mean operating time; mean repair time; mean data access rates; or similar performance and cost metrics. Some of these variables may be a single value variable, still others may be set or matrix variables. In some embodiments, the system may evaluate or calculate one or more storage related metrics as described in the commonly assigned U.S. patent application Ser. No. 11/120,662, now U.S. Pat. No. 7,346,751, entitled “SYSTEMS AND METHODS FOR GENERATING A STORAGE-RELATED METRIC”, U.S. application Ser. No. 11/639,830, filed Dec. 15, 2006, entitled “System and method for allocation of organizational resources”, U.S. application Ser. No. 11/825,283, filed Jul. 5, 2007, entitled “System and method for allocation of organizational resources”, which are hereby incorporated herein in their entirety, which is hereby incorporated by reference in its entirety. Such storage metrics may also be utilized as variables within a cost function.

The system may evaluate a cost function as follows. First, the system may mathematically transform the cost function variables to create a second set of intermediate variables (e.g., to normalize the variables). Each variable may be subjected to a different transformation. The transformations may be a linear transformation (including an identity transformation) or non-linear transformation. The transformations may also be invertible or non-invertible transformations. Non-exhaustive examples of transformations include:

- scaling the variable (by a constant);
- raising the variable to a power;
- taking a logarithm of the variable;
- applying a ceiling or floor mapping to the variable (i.e., quantization);
- reducing a set variable to its mean value, variance or other moment.

The transformation applied to a cost function variable may also merge a number of these suitable transformations. Second, the system may evaluate the cost function by mathematically combining the various intermediate variables. The combination may be a linear combination or a non-linear combination. Non-exclusive examples of combinations include any polynomial of the intermediate variables, including a simple summation of the various intermediate variables. Often, a cost function is a weighted summation of various cost function variables.

The system evaluates the same cost function for each storage device candidate and each group of data objects. However in other embodiments, the system may utilize different cost functions for different groups of data objects. In still other embodiments, the system may utilize different cost functions for different types of storage devices (e.g., there may be one cost function for optical media devices, another for tape media devices, and yet another for cloud storage sites). The cost function(s) and their associations with particular groups or storage media types may be defined in the storage policy or elsewhere.

80

At block **2735**, the system compares the costs associated with the various candidate storage devices. For example, the system compares these various costs to identify one or more candidates (“identified devices” or “sites”) having an associated cost that is lower than the other candidates. If more than one storage site is identified, the system may divide the group of data into one or more subgroups, and associate each with an identified site. However, in some embodiments, the system may also compare these costs to make other types of determinations. For example, the system may select identified sites using criteria other than minimizing associated cost. As another example, the system may compare the costs to ensure that at least one candidate satisfies a particular criteria, such having an associated cost that falls below a specified maximum value (that may be defined in the storage policy). Depending on the results of these determinations, the system may repeat some or all of blocks **2710-2735** using different quote parameters, different groupings, and/or different cost functions and/or may take other actions such as notifying an administrator. For example, in some embodiments, the system may repeat block **2715** by making another round of quote requests to some cloud storage sites that includes lower suggested or maximum rates (counteroffers to the first set of quotes).

At block **2740**, the system may transmit instructions to the jobs agent **220** (or other component) regarding the identified storage location of the group of data objects (or if the group has been subdivided, the identified storage location of each subgroup of data objects). For example, the system transmits instructions to the jobs agent **220** to migrate or transfer the data objects of the group or subgroup to its identified storage location. In some embodiments, the system may also transmit other information to the jobs agent **220** regarding the migration/transfer of the data objects. For example, the system may transmit a token or other identifier associated with a winning quote and/or may transmit information regarding the schedule of data migration/transfer. In some embodiments, the system may instead instruct a secondary storage computing device **265** or other system component regarding the identified storage location of a group or subgroup of data objects.

29. Process for Scheduling Cloud Storage Requests

FIG. **28** is a flow diagram illustrating a process **2800** for scheduling cloud storage requests received from auction clients; the process **2800** may be performed by an auction service component (not shown) forming part of a cloud storage site **115A-N** or any other suitable system (e.g., a component of a cloud storage brokerage site). An auction client may be a component of a storage manager **105**, a secondary storage computing device **165**, or any other device seeking cloud storage. For simplicity, the process refers to requests for an upload of data from an auction client (or related device) to a cloud storage site **115A-N**; however, auction clients may make requests for any type of cloud storage operation that requires system resources from a cloud storage site (e.g., downloading data or searching the contents of stored data).

In this process **2800**, the auction service evaluates requests from auction clients to upload data to the cloud storage site. The auction service may respond to some or all auction clients with a quote for their requested upload (“a quoted job”). Those requests that do not receive a quote in response may be queued for additional evaluation later (“queued requests”). If a quote is accepted by an auction client, the upload may be added to a list of “scheduled jobs.” Once a job is scheduled, other components within the cloud

US 10,248,657 B2

81

storage site (e.g., file servers) may accept the associated upload during its scheduled upload window.

The process **2800** begins at block **2805**, when the auction service determines the current system capacity and applicable quotation policies. In particular, auction service may access capacity policies, scheduled or quoted jobs, queued requests, quotation policies, and/or other information about system capacity and pricing. A “capacity policy” is generally a data structure or other information source that includes a set of preferences and other criteria associated with allocating system resources. The preferences and criteria may include, the system resources (e.g., data transfer volume or bandwidth) available for auction during specified periods, scheduled maintenance windows, and the current storage capacity available on particular servers or devices. The auction service may also determine the system resources required for jobs already scheduled or quoted. Using this information, the auction service may determine the available system resources available for providing new quotations.

The auction service may also access a quotation policy. A “quotation policy” is generally a data structure or other information source that includes a set of preferences and other criteria associated with generating a quote in response to auction client requests. The preferences and criteria may include, but are not limited to: a revenue function; a pricing function; pricing rate tables; codes and schedules associated with marketing promotions; a list of preferred and/or disfavored auction clients; current system capacity; classes or quality of storage; retention policies; upload time periods; data characteristics; compression or encryption requirements; the estimated or historic cost of storage, including the cost of power. A “revenue function” is generally a description of how the auction service may numerically evaluate the projected revenue (and/or other benefits) that would be generated by one or more auction client requests. A “pricing function” is generally a description of how the auction service may generate the various values (e.g., pricing rates) associated with a responsive quote.

At block **2810**, the auction service may receive one or more new requests from auction clients seeking cloud storage. The request may include various information such as: a unique identifier that the auction client has associated with the request; an identifier associated with a prior request made or a quote received from the site (e.g., in the case of a counter offer); information that identifies the auction client making the request (or identifies a related entity, such as a billing party); the amount of storage space desired; how the data will be accessed once stored (e.g., accessibility of data, including desired data transfer rates); suggested or required upload window; estimated storage lifetime of data; the type of storage medium desired (e.g., tape or optical or magnetic media); suggested download, upload, and/or storage pricing rates (and/or a promotional code or similar indicator of a pricing rate package); and/or any other information suitable for requesting cloud storage. The format and content of the request will typically conform to a specified API or similar convention employed by the auction service.

Although not shown, during block **2810**, the auction service may authenticate each of the requests and/or auction clients to ensure that each request is from a valid auction client. This authentication may happen via any acceptable method, including the use of passwords or security certificates. Those requests that cannot be authenticated may be discarded by the auction service without further consideration.

At block **2815**, the auction service evaluates queued and new requests (collectively the “pending requests”) and gen-

82

erates responsive quotes. To do so, the auction service may first identify those requests that either (1) do not satisfy minimum requirements specified by the quotation policy, or (2) cannot be accommodated due to a lack of system resources. Typically, the auction service will reject such requests by removing them from the list of pending requests. However, the auction service may also (1) send a quote with terms different from those requested (e.g., with higher rates or with a different scheduled upload window) in order to conform to the quotation policy, (2) send an explicit rejection of the request to the auction client, (3) queue the request for later evaluation, and/or (4) take another appropriate action.

At **2815**, the auction service may next identify which remaining pending requests should receive quotes and generate quotes. The auction service will apply the preferences and criteria specified in the quotation policy described previously to determine which “winning” requests should receive responsive quotes. In some embodiments, the auction service will choose the set of requests that results in a maximum combined value of a revenue function. Those pending requests that do not receive quotes will typically be queued by the auction service for later evaluation, but the auction service may also (1) send an explicit rejection of a request to the auction client, (2) remove it from the list of pending requests, and/or (3) take another appropriate action.

For each winning request, the auction service will generate a responsive quote. Quotes generated may specify: the unique identifier that the auction client has associated with the request; various pricing rates for different types of data operations (e.g., a first rate for an initial upload to the site, a second rate for downloads from the site, and a third rate for searching or accessing the data, a fourth rate for continued storage and maintenance of the data on the site (e.g., a rate charged for each gigabyte stored per month)); maximum storage space allotted; maximum or minimum storage lifetime; the accessibility of stored data; time windows during which data may be transmitted to the site or retrieved; etc. Each quote will typically include a token or other identifier associated with the quote and may specify an acceptance window during which the quotation will be honored by the site. The auction service generally applies the preferences and criteria specified in the quotation policy described previously (including a pricing function) to determine the values given in the quotes. For example, the pricing function may require the auction service to specify upload and storage rates associated with a marketing promotion, even if the client request proposed higher pricing rates. However, in some embodiments, the auction service may simply utilize in its quote some or all of the values proposed in the request.

At block **2820**, the auction service sends a copy of the generated quotes to auction clients. In response, each auction client may send another request (e.g. a “counteroffer”), may send an indication of acceptance of the quote and/or may take no action in response.

At block **2825**, the auction service may receive an indication of acceptance of one or more quotes. For each accepted quote, the auction service may add the associated upload to the list of scheduled jobs so that other system components will accept the upload. For example, the auction service only adds an upload to the list of scheduled jobs if the acceptance is received within the specified acceptance window. If the acceptance is received outside of this window, the auction service may treat the acceptance as it would a new request and repeat some or all of the previous blocks.

US 10,248,657 B2

83

30. Process for Encrypting Files within Cloud Storage

As described previously with respect to FIG. 3B, when a system migrates or copies data to secondary storage, including secondary cloud storage, the system may encrypt the data before or after a secondary copy or archival copy is created. When data is encrypted prior to migrating or copying data to secondary storage, the encryption enhances the “at-rest” security of files stored within a cloud storage site 115A-N, by reducing the risk of unauthorized access to the files’ content. In such implementations, it may be desirable to store encryption keys (and/or other information necessary to decrypt files) within the storage operation cell 150, not within the cloud storage site 115A-N used to store the encrypted files. In this way, even an operator of a cloud storage site may not breach the security of an encrypted file. If local encryption occurs within the storage operation cell 150 prior to copying or migrating data to a cloud storage site 115A-N, the encryption keys or similar encryption information may easily be stored within storage operation cell (e.g., within a local index or database of the storage operation cell or a different storage device 115). Alternatively, if local encryption is performed within a storage operation cell 150, the storage operation cell 150 may “scramble” encryption keys and store the scrambled keys with the encrypted files. This method provides some level of protection against intrusions, even intrusions by the operator of a cloud storage site. Further details may be found in U.S. Patent Publication No. US2008-0320319A1 referenced above.

In some circumstances, however, decrypted files may be stored within a cloud storage site 115A-N without first encrypting the files within the storage operation cell 150. In such circumstances, it may be desirable to later encrypt the files stored on the cloud storage site to protect those files thereafter.

FIG. 29 illustrates a process 2900 for encrypting files stored within a cloud storage site 115A-N. The process may be performed by cloud storage submodule 236, or any other suitable system component. The process begins at block 2910, when cloud storage submodule 236 receives a request to encrypt a file located on a target cloud storage site. For example, cloud storage submodule 236 may receive an indication of which target files within a target cloud storage site should be encrypted. Cloud storage submodule 236 may also receive an indication of which encryption method should be utilized, one or more encryption keys and/or additional information.

At block 2915, cloud storage submodule 236 determines if the type of encryption method requested is supported by the API provided by the operator of the target cloud storage site 115A-N. If it is not, the process proceeds to block 2940. Otherwise, the process 2900 proceeds to block 2930, where cloud storage submodule utilizes the mapping described herein to generate vendor-specific API calls to encrypt the original file. The process then returns.

If the target cloud storage site API does not support the desired type of encryption, the process 2900 proceeds instead to block 2940. At block 2940, cloud storage submodule 236 utilizes its mapping described herein to generate and send a vendor-specific API call to download the file to the cloud storage submodule, or another component of the storage operation cell 150. At block 2945, the downloaded file is encrypted locally (e.g., by a component of storage operation cell 150 configured to perform encryption, such as a secondary storage computing device 165). At block 2950 cloud storage submodule utilizes its mapping described herein to generate and send vendor-specific API calls to overwrite the original file with an encrypted version. For

84

example, cloud storage submodule may utilize vendor-specific API calls that open the original file for writing, write the contents of the encrypted version of the file to the original file, and close the original file. Alternatively, cloud storage submodule 236 may utilize vendor-specific API calls to create a new file on the target cloud storage site 115A-N, write the contents of the encrypted version of the original file to the new file, close the new file, and delete the original file.

31. Protecting Remote Office and Branch Office (Robo) Data

In one example, the systems described herein may be utilized to protect remote office and branch office (ROBO) data. In some implementations, a subset of clients 130 may be “remote clients” who are geographically separated from other components of an associated storage operation cell 150. Remote clients 130 may only be connected to other components of an associated storage operation cell 150 via a WAN such as the Internet due to a physical separation between the remote client 130 and other system components. One intuitive example of a remote client 130 is a laptop computer utilized by a traveling employee: when the employee is traveling, she will be geographically separated from their company’s main storage operation cell 150.

In such implementations, a remote client 130 may include a media file system agent 240, including a cloud storage submodule 236, to permit data agents 195 on the remote client to directly write data to a cloud storage site 115A-N (e.g., over a network connection established by an HTTP client subagent). For example, in this manner a remote client 130 may directly mirror data to cloud-based storage for disaster recovery purposes and/or to comply with other system-level data retention policies. In accordance with system-wide storage and scheduling policies, other system components (e.g., jobs agent 220) may instruct a remote client 130 regarding when and how to perform a remote storage operation. Additionally, a remote client 130 may provide information regarding a storage operation made in this manner to other system components, so that those system components may update the various system-wide indices and databases to reflect the storage operation. For example, client 130 may provide storage manager 105 with information that is sufficient for storage manager 105 to update management index 211, management light index 245, SS index 261, SS light index 247, and deduplication database 297.

In such implementations, the system may avoid routing data slated for cloud storage through a secondary storage computing device 165, thereby conserving system resources (e.g., the bandwidth of a secondary storage computing device). Such implementations preserve the ability of the storage cell 150 to perform upon all data, including data generated by remote clients 130: policy-driven storage, ILM, content indexing, data restoration, and searching.

In some implementations, a group of clients 130 may be geographically separated from most of the system components of an associated storage operation cell 150 but may not be geographically separated from one or more locally accessible secondary storage computing devices 165. For example, a group of clients (e.g. a group of clients associated with a particular branch office of a company) may be connected to a locally accessible secondary storage computing device 165 over a LAN, but may be connected to other components (e.g. storage manager 105, storage devices 115, other secondary storage computing devices 165) only over a WAN like the Internet. In such implementations, the group of clients 130 may copy or migrate data to a locally accessible secondary storage computing device,

US 10,248,657 B2

85

which may in turn write this data to a cloud storage site 115A-N in accordance with applicable system-wide storage and scheduling policies.

Thus the locally accessible secondary storage computing device 165 may mirror data from a branch office directly to cloud-based storage for disaster recovery purposes and/or to comply with other data retention policies, without first routing that data over a WAN to other system components. Additionally, a locally accessible secondary storage computing device 165 may provide information regarding a storage operation made in this manner to other system components, so that those system components may update the various system-wide indices and databases to reflect the storage operation. For example, a locally accessible secondary storage computing device 165 may provide storage manager 105 with information that is sufficient for storage manager 105 to update management index 211, management light index 245, SS index 261, SS light index 247, and deduplication database 297. Such implementations preserve the ability of the storage cell 150 to perform upon all data, including data generated by remote clients 130: policy-driven storage, ILM, content indexing, data restoration, and searching.

Alternatively or additionally, a group of clients may be connected to a locally accessible cloud gateway 1540 over a LAN, but may be connected to other system components only over a WAN. In such implementations, the locally accessible cloud gateway 1540 may provide the same functionality of a locally accessible secondary storage computing device 165 described in this section, in addition to other cloud gateway functionality described herein.

32. Conclusion

IT organizations continue to deal with massive unstructured data growth, stronger regulatory requirements and reduced budgets. To meet the needs of more stringent data retention requirements and faster RTO's, many users have over provisioned low-cost disk storage which, combined with non-integrated data management products, creates inefficient storage infrastructures resulting in high operating costs. In fact, many data centers have reached a limit where there is no power or real estate left to continue expanding.

Today's IT organizations are struggling to keep pace with multiple factors that are starting to severely impact the ways that they protect, manage and recover their business-critical data, data that is increasingly located in remote offices and on user laptops/desktops, outside of core IT facilities. Relentless, ongoing data growth across the enterprise, often growing at 30-50% per year ensures that some storage teams are looking at a doubling of capacity requirements every 18 months. Increased government regulation around data retention policies adds to the burden, often requiring that critical data be kept for years or even decades. Further, many IT organizations worldwide are being forced to justify not only incremental spending, but also justify their existing expenses and/or headcount in the face of potential budget cuts.

Cloud storage sites represent an increasingly viable option to manage the growing bodies of data. They promise lower costs through better utilization and management of the underlying storage infrastructure. Cloud-based storage also eliminates the need to buy lots of spare capacity in anticipation of future storage growth, enabling companies to "pay as you grow". Further cloud-based storage enables IT organizations to minimize investment in new Data Center capacity, and extends the life of their existing investment in both building and computing infrastructure.

86

However leveraging cloud-based storage can be challenging for some organizations for a variety of reasons. First is the inherent complexity associated with managing two sets of infrastructure, one physical and on-premise and another online in the virtual storage cloud. This duplication of effort extends across a number of crucial aspects of data management including: Backup, Archive, Reporting and search/eDiscovery. There are challenges often associated with taking full-advantage of cloud-based storage. The first is complexity associated with moving data into and out of the cloud. Gateway appliances are often expensive, complex and represent a short-term fix that can aggravate infrastructure management challenges as the use of cloud-based storage grows. A related concern is the amount of data being moved to and managed within cloud storage. This not only impacts the ongoing service charges, which are often priced on a per-GB basis but also impacts the ability to meet backup windows over limited bandwidth. Data security and reliability are critical both from a data integrity perspective as well as to ensure that a company's critical data is not accessed by unauthorized parties, even including individuals working for a cloud-storage provider. Further, companies don't want to be locked in to a single vendor when it comes to data stored in the cloud. So data portability becomes critical, along with the ability to choose from among a variety of providers for specific performance and pricing requirements.

The systems herein permit policy-driven storage that defines what data stays on-premise and what moves to the cloud. Storage policies may consider "data value" determined from factors such as (a) access requirements, (b) latency requirements, and (c) corporate requirements including: how recently was the data accessed, how often was the data required over a given time period, such as the last 12 months, how many end-users/applications required access to the data in the last 12 months, how quickly will the data need to be restored, what downstream applications/processing are dependent on the data, whether the data needs to be identified and pulled in/put on Legal Hold for an eDiscovery request, whether the data contains corporate trade secrets or IP, whether the data might be considered highly sensitive (e.g., legal communication, or social security numbers).

The systems and methods described herein provide integrated data management platforms that address a wide variety of data management needs. The systems and methods herein may deliver unified data management from a single console. When combined with cloud storage, a seemingly unlimited storage pool, these systems and methods may offer users lower operating costs, ensure disaster recovery, while improving long-term compliance management.

The systems described herein provide a unified data management platform that may be built on a single codebase or as a unified application, with modules or agents for backup and recovery, archive, replication, reporting, and search/eDiscovery. These systems may provide automated, policy-based data movement from local, deduplicated copies into and out of cloud storage environments—all from the same centralized console. This incremental approach to data management may permit organizations to leverage the economics of cloud-based storage.

The systems and methods described herein may result in various other performance advantages. For example, these systems and methods may reduce administrative and storage overhead for infrequently-accessed data in a data center by automatically tiering older/infrequently-accessed data in a data center to more efficient, lower-cost cloud-based storage, freeing up existing capacity to accommodate ongoing data growth.

US 10,248,657 B2

87

Integrated deduplication ensures that unique (or semi-unique) data segments are stored “in the cloud”, minimizing costs associated with redundant data across backups and archive. Block-based data deduplication and replication reduce network bandwidth requirements to minimize network costs and backup windows. Deduplication also reduces ongoing storage costs up to 75%, minimizing operational expenses across the entire lifespan of the data being retained.

The systems described herein may permit a better data encryption approach to meet applicable requirements. A user may protect data starting from the source with in-stream encryption, and then extend encryption to data “at-rest”. This ensures that not only is a user protected during data migration, but also from unwarranted access of data already on the cloud. Because the data encryptions are controlled by a company’s IT team, data is safe even from unintentional access by a cloud storage providers’ IT staff.

By providing encryption of data in-flight and at-rest data, the systems and methods help protect data, even from cloud storage site operators. Built-in data encryption and verification technology ensures data has been securely and safely written to the cloud without errors. Encryption of data at-rest helps ensure that only appropriate personnel have full access to readable data, no matter where it’s stored.

The systems herein are designed to work with a wide variety of storage partners, both physical and a growing number of cloud-based storage providers. Today these include Amazon’s S3, Microsoft Azure, Nirvanix SDN with upcoming support for Iron Mountain and Rackspace. This open approach ensures that additional cloud-storage vendors will continue to be added in the future to increase the choices available.

The systems described herein may deliver a seamless solution for data-aware movement into cloud storage to help reduce overall complexity and costs. Lack of a native cloud-storage connector often requires complex scripting, adding both time and risk to moving data into the cloud. Using gateway appliances can present an ongoing and growing management burden as cloud-storage use increases. An integrated approach such as that described herein eliminates the costs and risk associated with either approach. Integrated data management of both local storage and cloud storage from a single console minimizes administrative overhead and the need for specialized gateway appliances. The systems described may also be readily configured to support an expanding list of industry-leading cloud providers to provide flexibility and choice for how to host cloud-based data immediately and in the future. Native integration with REST/HTTP protocols seamlessly extends data management to the cloud without the need for scripting or specialized vendor-specific gateway appliances.

A highly efficient platform automates the movement of data across systems from a variety of storage vendors, and across different types of storage devices including disk, tape, CAS, VTL, optical—and now cloud storage. By integrating these functions together, users can leverage one interface to manage one data management suite across a virtual shared storage environment. Moving data into and out of the cloud using the systems herein is as easy as moving data between any 2 data storage tiers. For existing users, this can be done in as little as 3 steps: choosing one or more cloud-storage sites, setting up a storage service similar to what a user would do to add disk-based storage, and adding the new cloud-based storage to existing backup and/or archive policies and data paths.

As data management expands to beyond a physical infrastructure, and into the cloud, legal and reporting require-

88

ments continue to grow as well. The systems described herein may offer at least four key benefits for search/eDiscovery:

1. Indexes of all data retained can be kept on-premise. This enables a user to retain control of the most critical and sensitive aspects of information management, and ensures that content indexes are accessible only to designated personnel within an organization.

2. Since the indexes are searchable locally, there is no latency with regards to data that may be retained in the cloud over a number of years or even decades. This reduces the amount of time and data required by a company’s legal and/or IT teams.

3. Only the specific data required for eDiscovery requests is restored back from the cloud. This saves on bandwidth, the time needed for data restore and minimizes the data retrieval costs charged by a cloud-storage vendor.

4. Global indexing of all relevant data, from the Data Center to remote sites, mobile users and cloud-based data. This ensures that a company has a global view of all their data, so that a company can also avoid the legal and financial risks associated with incomplete responses to eDiscovery requests.

Integrated content indexing done prior to tiering to the cloud, ensures that administrators can do fast searches on a local index and retrieve only specific data that meets the search criteria.

A variety of data reduction techniques can also be used to minimize the amount of data sent to the cloud, and minimize the cloud-based capacity usage. Block-based deduplication reduces backup and archive times and data volumes by filtering out redundant data before it reaches the cloud. This can be done in a data center or even at remote sites, depending on the system configuration. Additional data management approaches such as incremental backups and data compression at the source can further reduce the amount of data in-transit and at-rest.

As data volumes continue to increase, many companies find themselves bumping up against the capacity, cooling or power limitations of their existing data centers. Meanwhile they’re now required to keep every-growing amount of data as mandated by their corporate legal staff, acting under the aegis of governmental regulation. This 3-way balancing act between capacity, compliance and cost requires a flexible approach to data management that requires a multi-tier approach that extends to cloud-based storage. The systems described herein may be used for an end-to-end approach to tiering a combination of data from within the data center, from remote offices and from individual employees worldwide.

A second use case of the described systems centers around protecting data outside of the Data Center and storing it in the cloud. This enables the central IT team to control the movement and management of data along with defining the appropriate data retention and recovery policies.

Data from remote offices (and even end-users/employees if configured) can be backed up directly to cloud-based storage, eliminating the need to migrate the data to the data center first, and then migrating the data again to the cloud. In other cases, data may be mirrored to cloud-based storage for Disaster Recovery purposes as well for long-term data retention. As data ages past retention requirements it can be automatically deleted in the cloud, creating ongoing savings in capacity utilization charges.

Because data is managed just the same as if were stored in a core data center, Storage Reporting and Management (SRM) can be easily used to monitor, analyze and monitor

US 10,248,657 B2

89

data across the enterprise regardless of whether it stored in the cloud, in a core data center or in remote offices or other locations.

The systems and methods described herein may provide the following benefits and features, inter alia:

Ensuring data security when: data is in transit, both to and from the cloud and when data is at-rest (including security from service-provider personnel).

Portability, by permitting a user to easily move data back from the cloud if required, and to move data quickly between cloud-based storage providers, to improve price and performance.

Restoring data quickly and directly from any physical or cloud-based storage tier.

Configuring data management policies so that most frequently accessed data is more easily and quickly retrieved when required.

Matching network bandwidth capacities to data's RTO (recovery time objective) requirements.

Archiving data to the cloud, including setting up automated retention and deletion policies.

Easily configurable global reporting of all data (physical and in-the-cloud).

Easily and securely extending cloud-based data management to include search/eDiscovery.

Unless the context clearly requires otherwise, throughout the detailed description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense (i.e., to say, in the sense of "including, but not limited to"), as opposed to an exclusive or exhaustive sense. As used herein, the terms "connected," "coupled," or any variant thereof means any connection or coupling, either direct or indirect, between two or more elements. Such a coupling or connection between the elements can be physical, logical, or a combination thereof. Additionally, the words "herein," "above," "below," and words of similar import, when used in this application, refer to this application as a whole and not to any particular portions of this application. Where the context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number respectively. The word "or," in reference to a list of two or more items, covers all of the following interpretations of the word: any of the items in the list, all of the items in the list, and any combination of the items in the list.

The above Detailed Description of examples of the invention is not intended to be exhaustive or to limit the invention to the precise form disclosed above. While specific examples for the invention are described above for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. While processes or blocks are presented in a given order in this application, alternative implementations may perform routines having blocks or steps performed in a different order, or employ systems having blocks in a different order. Some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified to provide alternative or subcombinations. Also, while processes or blocks are at times shown as being performed in series, these processes or blocks may instead be performed or implemented in parallel, or may be performed at different times. Further any specific numbers noted herein are only examples. It is understood that alternative implementations may employ differing values or ranges.

The various illustrations and teachings provided herein can also be applied to systems other than the system described above. The elements and acts of the various

90

examples described above can be combined to provide further implementations of the invention. Some alternative implementations of the invention may include not only additional elements to those implementations noted above, but also may include fewer elements.

All patents and applications and other references noted above, including any that may be listed in accompanying filing papers, are incorporated herein by reference in their entireties. Aspects of the invention can be modified, if necessary, to employ the systems, functions, and concepts included in such references to provide further implementations of the invention.

These and other changes can be made to the invention in light of the above Detailed Description. While the above description describes certain examples of the invention, and describes the best mode contemplated, no matter how detailed the above appears in text, the invention can be practiced in many ways. Details of the system may vary considerably in its specific implementation, while still being encompassed by the invention disclosed herein. As noted above, particular terminology used when describing certain features or aspects of the invention should not be taken to imply that the terminology is being redefined herein to be restricted to any specific characteristics, features, or aspects of the invention with which that terminology is associated. In general, the terms used in the following claims should not be construed to limit the invention to the specific examples disclosed in the specification, unless the above Detailed Description section explicitly defines such terms. Accordingly, the actual scope of the invention encompasses not only the disclosed examples, but also all equivalent ways of practicing or implementing the invention under the claims.

We claim:

1. A method for storing a secondary copy, of an original data set, on a cloud storage site using a cloud gateway, wherein the cloud gateway is coupled between multiple computers and one or more cloud storage sites via a network, the method comprising:

identifying data blocks within a cache of the cloud gateway that satisfy certain criteria,

wherein the original data set comprises data blocks, and wherein the certain criteria are from a storage policy;

performing block-level deduplication of the identified data blocks to create a deduplicated set of data,

wherein the block-level deduplication includes—

determining a size for a container file to utilize when

deduplicating the identified data blocks; and

deduplicating at least some of the identified data

blocks to create one or more container files con-

taining deduplicated data,

wherein at least one of the container files has the determined size; and

storing the deduplicated set of data on the cloud storage site by:

buffering data, to a data buffer, for transmission to the cloud storage site;

repeating the following steps while the data buffer is not full:

receiving a file system request to write a group of

data to the cloud storage site; and

adding the group of data to the buffer;

converting a file system request to one or more application program interface calls associated with the cloud storage site; and

transmitting contents of the data buffer to the cloud storage site using the one or more application program interface calls associated with the cloud storage site.

US 10,248,657 B2

91

2. The method of claim 1, further comprising identifying the cloud storage site on which to store the secondary copy of the original data set by:

- identifying two or more candidate cloud storage sites;
- accessing a storage policy having a set of preferences and storage criteria, 5
- wherein the set of preferences and storage criteria includes at least two of the following:
 - one or more preferred cloud storage sites,
 - one or more preferred classes or quality of cloud storage sites,
 - requirements regarding deduplication of the original data set,
 - requirements regarding encryption of the original data set, 15
 - requirements regarding compression of the original data set,
 - quality of a network connection available to the cloud storage site, 20
 - one or more data retention periods,
 - data characteristics of at least some data in the original data set,
 - estimated or historic usage associated with operating one or more system components, 25
 - frequency with which the original data set was accessed or modified during a particular time period,
 - a specified level of fault tolerance, or
 - one or more geographical locations or political states in which data storage devices for a cloud storage site exist; and
- selecting at least one of the two or more of the candidate cloud storage sites based at least in part on the set of preferences and storage criteria in the storage policy. 35

3. The method of claim 1 wherein the contents of the data buffer are transmitted to the cloud storage site using at least one of hypertext transfer protocol (HTTP) and HTTP over Transport Layer Security/Secure Sockets Layer.

4. The method of claim 1 wherein the certain criteria 40 include time-based criteria.

5. A system for creating a secondary copy of an original data set using a cloud storage site, the system comprising a memory and processor that are configured to:

- identify sub-objects of the original data set that satisfy certain criteria, 45
- wherein the certain criteria are related a storage policy, and
- wherein the original data set is received from one or more client computers; 50
- perform deduplication of the identified data sub-objects to create a deduplicated set of data; and,
- forward the deduplicated set of data to the cloud storage site, wherein the forwarding includes:
 - converting file system requests into application program interface calls associated with the cloud storage site; and, 55
 - forwarding the data to the cloud storage site using the one or more application program interface calls associated with the cloud storage site.

6. The system of claim 5, wherein the memory and processor are further configured to:

- determine a size for a container file and for deduplicating at least some of the data sub-objects to create one or more container files containing deduplicated data, 65
- wherein at least one of the container files has the determined size.

92

7. The system of claim 5, wherein the forwarding further includes:

- buffering data, to a data buffer, for transmission to the cloud storage site by:
 - receiving a file system request to write a group of data to the cloud storage site; and
 - adding the group of data to the data buffer.
- 8. The system of claim 5, wherein the certain criteria include time-based criteria, wherein the deduplication includes block-level deduplication, and wherein the block-level deduplication includes—
 - determining a size for a container file to utilize when deduplicating the identified data blocks; and
 - deduplicating at least some of the identified data blocks to create one or more container files containing deduplicated data,
 - wherein at least one of the container files has the determined size; and
 - wherein the container file is forwarded to the cloud storage site.
- 9. The system of claim 5, wherein the forwarding further includes:
 - buffering data, to a data buffer, for transmission to the cloud storage site by repeating the following steps while the data buffer is not full:
 - receiving a file system request to write a group of data to the cloud storage site; and
 - adding the group of data to the buffer.
- 10. A computer-implemented method for copying multiple files at a cloud storage site, wherein the cloud storage site is coupled to a computer executing a file system for accessing a secondary storage computing device, the method comprising:
 - receiving a copy operation request to copy n number of files at the cloud storage site,
 - wherein each of the n number of files includes metadata and data, and
 - wherein the n number of files exceeds a threshold;
 - establishing a container size determined by one or more factors
 - processing the n number of files by—
 - copying the metadata of each of the n number of files to a first container;
 - copying at least a portion of the data for the n number of files into a second container, wherein the second container is separate from the first container; and
 - updating a data structure, wherein the data structure—
 - tracks, for each of the n number of files, a location of the metadata for that file in the first container, and
 - tracks, for the at least a portion of the data for the n number of files, a location of the data in the second container.
- 11. The computer-implemented method of claim 10 wherein the threshold is a number of files that the file system can operate on without system degradation.
- 12. The computer-implemented method of claim 10 wherein the threshold is related to at least one of the factors.
- 13. The computer-implemented method of claim 10 wherein the factors include at least one of:
 - a latency associated with a network connection to the cloud storage site, or
 - a bandwidth associated with a network connection to the cloud storage site, or
 - whether the cloud storage site imposes a restriction on a namespace associated with the computer or the file system, or

US 10,248,657 B2

93

whether the cloud storage site permits sparsification of data files, or
 a pricing structure associated with the cloud storage site, or
 a maximum specified container file size, or
 a minimum specified container file size.

14. The computer-implemented method of claim 10 wherein the size of at least one of the first and second containers is no greater than the established container size.

15. A tangible computer-readable storage medium whose contents cause a data storage system to perform a method of migrating data from local primary storage to secondary storage located on a remote cloud storage site, the method comprising:

identifying no more than $n-1$ data blocks, located within the local primary storage, that satisfy a criteria, wherein the $n-1$ data blocks represent a portion of a data file consisting of n blocks, and wherein the n blocks contain data written by a file system associated with the local primary storage; and determining a size for a container file in which to store some or all of the no more than $n-1$ data blocks; transferring data contained by the identified no more than $n-1$ data blocks from the primary storage to the secondary storage located on a cloud storage site, wherein transferring data includes writing data first to a container file of the determined size; and updating an index with information associating the transferred data with information identifying blocks within the secondary storage that contain the transferred data, wherein the information includes at least one uniform resource locator or logical address that identifies at least one logical location from which the transferred data may be accessed.

16. The tangible computer-readable storage medium of claim 15 wherein the index further comprises information associating the transferred data with information identifying tape offsets for secondary storage that contain the transferred data.

17. The tangible computer-readable storage medium of claim 15, further comprising:

receiving a copy operation request to copy m number of files at the cloud storage site, wherein each of the m number of files includes meta-data and data, and wherein the m number of files exceeds a size threshold.

18. The tangible computer-readable storage medium of claim 15, wherein determining the size for the container file considers:

a latency associated with a network connection to the secondary storage computing device; or
 a bandwidth associated with a network connection to the secondary storage computing device.

19. The tangible computer-readable storage medium of claim 15, wherein determining the size for the container file considers:

whether the cloud storage site imposes a restriction on a namespace associated with the computer or the file system; or

94

whether the cloud storage site permits sparsification of data files.

20. The tangible computer-readable storage medium of claim 15, wherein determining the size for the container file considers:

a pricing structure associated with the cloud storage site; a maximum specified container file size; or
 a minimum specified container file size.

21. A system for storing, on a cloud storage site, a secondary copy of an original data set, the system comprising:

at least one processor;
 memory coupled to the at least one processor, wherein the memory stores contents that, when executed by the at least one processor performs a method of:
 identifying a cloud storage site on which to store a secondary copy of a primary data set;
 updating an index of content to reflect at least some data content in the primary data set;
 deduplicating at least some of the data content in the primary data set;
 creating one or more container files containing the deduplicated data; and
 transferring the one or more container files to the cloud storage site,

wherein the transferring includes:

converting file system requests into application program interface calls associated with the cloud storage site; and,

forwarding the one or more container files to the cloud storage site using one or more application program interface calls associated with the cloud storage site.

22. The system of claim 21 wherein the transferring further includes:

buffering the one or more container files, to a data buffer, for transmission to the cloud storage site by repeating the following steps while the data buffer is not full:
 receiving a file system request to write a group of data to the cloud storage site; and
 adding the group of data to the buffer.

23. The system of claim 21 wherein the memory and processor are further configured to:

determine a size for a container file based on one or more factors,

wherein the factors include at least one of:

a latency associated with a network connection to the secondary storage computing device, or
 a bandwidth associated with a network connection to the secondary storage computing device, or
 whether the cloud storage site permits sparsification of data files, or
 a pricing structure associated with the cloud storage site, or

a maximum specified container file size, or
 a minimum specified container file size; and

wherein at least one of the container files has the determined size.

* * * * *